教育部–IBM高校合作项目
精品课程系列教材



基于RUP的软件测试实践

姚登峰　主编
韩玉敏　邱云峰　黄海瑞　李娟　编著



清华大学出版社

教育部-IBM 精品课程教材

基于 RUP 的软件测试实践

姚登峰　主编
韩玉敏　邱云峰　黄海瑞　李　娟　编著

清　华　大　学　出　版　社
北　京

内 容 简 介

本书介绍了 RUP 的特点、原则和概念及 RUP 的四级测试(单元测试、集成测试、系统测试和验收测试)。在内容的选取上对基本知识的建立、基本技能的培养两方面有所侧重,让学生形成对 RUP 的整体理论框架的基本认识,为学生毕业后从事软件测试职业和在专业上的持续发展奠定基础。

本书适合作为高等学校计算机及相关专业的本专科生教材。

图书在版编目(CIP)数据

基于 RUP 的软件测试实践 / 姚登峰主编;韩玉敏等编著. —北京:清华大学出版社,2009.9
(高等学校计算机基础教育教材精选)
ISBN 978-7-302-20247-9

Ⅰ. 基… Ⅱ. ①姚… ②韩… Ⅲ. 软件工具—测试—高等学校—教材 Ⅳ. TP311.56

中国版本图书馆 CIP 数据核字(2009)第 082248 号

责任编辑:谢 琛 张为民
责任校对:李建庄
责任印制:

出版发行:清华大学出版社　　**地　址**:北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　编**:100084
社 总 机:010-62770175　　**邮　购**:010-62786544
投稿与读者服务:010-62776969,c-service@tup.tsinghua.edu.cn
质 量 反 馈:010-62772015,zhiliang@tup.tsinghua.edu.cn
印 刷 者:
装 订 者:
经　销:全国新华书店
开　本:185×260　　**印　张**:26.5　　**字　数**:624 千字
版　次:2009 年 9 月第 1 版　　**印　次**:2009 年 9 月第 1 次印刷
印　数:1～0000
定　价:0.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。联系电话:010-62770177 转 3103　　产品编号:-

出版说明

在教育部关于高等学校计算机基础教育三层次方案的指导下，我国高等学校的计算机基础教育事业蓬勃发展。经过多年的教学改革与实践，全国很多学校在计算机基础教育这一领域中积累了大量宝贵的经验，取得了许多可喜的成果。

随着科教兴国战略的实施以及社会信息化进程的加快，目前我国的高等教育事业正面临着新的发展机遇，但同时也必须面对新的挑战。这些都对高等学校的计算机基础教育提出了更高的要求。为了适应教学改革的需要，进一步推动我国高等学校计算机基础教育事业的发展，我们在全国各高等学校精心挖掘和遴选了一批经过教学实践检验的优秀的教学成果，编辑出版了这套教材。教材的选题范围涵盖了计算机基础教育的三个层次，包括面向各高校开设的计算机必修课、选修课以及与各类专业相结合的计算机课程。

为了保证出版质量，同时更好地适应教学需求，本套教材将采取开放的体系和滚动出版的方式（即成熟一本，出版一本，并保持不断更新），坚持宁缺毋滥的原则，力求反映我国高等学校计算机基础教育的最新成果，使本套丛书无论在技术质量上还是文字质量上均成为真正的“精选”。

清华大学出版社一直致力于计算机教育用书的出版工作，在计算机基础教育领域出版了许多优秀的教材。本套教材的出版将进一步丰富和扩大我社在这一领域的选题范围、层次和深度，以适应高校计算机基础教育课程层次化、多样化的趋势，从而更好地满足各学校由于条件、师资和生源水平、专业领域等的差异而产生的不同需求。我们热切期望全国广大教师能够积极参与到本套丛书的编写工作中来，把自己的教学成果与全国的同行们分享；同时也欢迎广大读者对本套教材提出宝贵意见，以便我们改进工作，为读者提供更好的服务。

我们的电子邮件地址是：jiaoh@tup.tsinghua.edu.cn。联系人：焦虹。

清华大学出版社

序

随着软件在各行各业的广泛应用，软件质量成为提高市场竞争力的重要因素。软件测试作为提高质量的主要方法之一，已经成为企业和用户关注的焦点。而专业化的软件测试外包服务已经成为国际软件产业的重要业务之一。特别是中国政府大力鼓励和推动外包服务产业的发展，因此，作为入门门槛比较低的软件测试外包业务急速发展，造成短时间内测试人才供应紧缺，特别是中高级软件测试管理人才更是供不应求。

软件测试是一项技能密集型的职业，需要软件测试人员掌握测试技术、流程、度量和管理等综合知识。2005 年 10 月 25 日，劳动部正式将计算机软件产品检验员（即软件测试工程师）列为第四批新职业。

从我多年教学经历中了解到大学生普遍对软件测试有偏见和误解。一些同学总是认为软件测试没有什么技术含量，认为搞不了软件开发的人才会去干测试，其实这是不对的，要成为一名优秀的软件测试工程师不仅需要深入了解软件工程知识，掌握扎实的软件测试理论，而且还要精通软件测试方法，精通软件测试工具和环境，精通行业领域的业务。在这些知识掌握的基础上还要积累丰富的测试实践经验。

2007 年 8 月，由教育部软件工程专业教学指导委员会、上海交通大学软件学院、清华大学出版社等组织的第一次软件测试教学研讨会在上海举行，旨在希望加强软件测试教学和实践的改进。要在大学形成完整的软件测试教育体系和实验环境还不是一件容易的事情，需要与产业界合作。

我们注意到，随着软件测试行业不断的发展，逐渐积累和提炼了许多最佳实践和案例，软件测试方法和工具也在不断发展和完善。例如，随着计算机网络技术和通信技术的发展，面向对象软件开发方法的不断改进，软件测试和 RUP 理论融合、基于 RUP 的软件测试正成为国内外研究的热点之一，相关的工具也已获得业界普遍应用。《基于 RUP 的软件测试实践》作为被列入 2007 年教育部-IBM 精品课程建设项目的配套教材得到学校、IBM 公司和出版社等各方面的大力支持。书中着重介绍了 IBM 公司提出的 RUP 思想和 RUP 测试工具的使用，为软件测试课程及其实验教学提供了有效的支持。本书还适用于业界软件测试入门者和初级测试人员，以及一些急于学习、更新软件测试知识和提高软件测试技术的读者。

我相信本书能促进测试人员技术水平的提高，也为更多的人群加入测试队伍提供实践向导，帮助他们了解最流行的软件测试技术和测试工具，学习最常见的、最常用的先进技术。希望本书能成为软件开发和软件测试工程师的良师益友。

最后，我还要告诉读者我为本书的作者姚登峰感到骄傲，从他在北大软件与微电子学院求学到走上工作岗位，他身残志坚，克服了重重困难，付出数倍于常人的努力，他在软件测试领域孜孜以求、不断进取的精神和取得的成果令人钦佩。衷心祝贺本书的出版，衷心祝福姚登峰能够取得更大的成就。

北京大学软件与微电子学院院长、博士生导师　陈钟

2009 年春

前言

本书是 2007 年度教育部-IBM 精品课程建设项目配套教材，得到了教育部和 IBM 中国公司的大力支持。本课程教材建设从实用角度出发，遵循 ACM IEEE-CS 计算机教学计划 2001，注意把握学生已有的知识背景和接受能力，通过对软件企业的调研，了解到用人单位对测试人才培养的意见、需求和建议，特别是对人才知识结构、能力结构和素质等方面的要求，结合我校“双证管理型”情况（双证是指在学生毕业时既有毕业证，又有 IBM 课程结业证书），我们将软件测试的教学与认证考试有机地结合起来，将《国家软件评测师认证考试大纲》所要求的考试内容有选择性地纳入到教学计划中来。合理地制定了教学大纲。使学生完成本课程后，能熟练掌握软件测试的基本理论和基本技能，为后续课程学习和以后工作奠定坚实的基础，保证教学目标和人才培养目标的双重实现。

本书以 RUP 测试过程为主线，一步步引导学生思索，探究软件开发和测试实践，通过参与项目使用软件测试工具对软件进行测试，使学生掌握岗位操作技能，在实践中理解并掌握知识。理论和实践教学在同一个项目上实现了统一，有利于学生自觉地应用理论知识解决实际问题。突出学生在教学过程中的主体地位。

现代工业心理学研究表明，高新技术工作岗位的工作人员所需要的知识，约一半是介于经验性知识和学科理论知识之间的一种特殊的知识，即“劳动过程知识”。软件测试在规范的软件生产中属于软件过程工程的重要组成，软件测试课程构建必须使学生能在“完整工作过程”中学习。同时注意引入在国际市场占有率高、有代表性的自动测试工具，如 Mercury、IBM 的有关产品，内容涵盖从安装到使用，并结合实际操作案例进行分析讲解，以此加深对概念和方法的理解，达到技术运用举一反三和知识传授与技能培养并重的目的。

按照 RUP 思想，RUP 确定了四级测试：单元测试、集成测试、系统测试和验收测试。这些测试级别可以是并列的，也可以是递进的，这取决于主测试计划（在项目级）和迭代测试计划（在迭代级）。本书还是按照单元测试、集成测试、系统测试和验收测试的顺序来排列。其中前面又加了基础部分，为学习 RUP 测试做准备。后面加了实践案例，运用所学知识开展实际测试。本书在软件测试课程内容的选取上对基本知识的建立、基本技能的培养两方面有所侧重，为学生毕业后从事软件测试职业和专业持续发展奠定基础。总的来说，基于 RUP 的软件测试实践课程的教学内容分为以下几个部分：

第 1 部分，软件测试基础：包括绪论、RUP 的基础理论、RUP 测试概论、手工测试与自动化测试等，即软件测试的起源和发展、测试行业的现状以及优秀测试工程师应该具备

的素质、RUP 的测试理论、自动化测试理论和实践、软件测试开发流程等。

第 2 部分，单元测试：包括测试管理、单元测试。主要讲解测试管理的不同的阶段：组织、计划、创作、执行以及报告；测试管理所面临的挑战；IBM 公司对测试管理所提的建议；实施测试管理自动化的原因；使用 TestManager 实现测试管理的自动化；IBM 公司的单元测试工具 Rational PurifyPlus。

第 3 部分，集成测试：包括组件测试和运行时分析解决方案，主要讲解组件的定义、组件测试的概念、进行组件测试的原因、组件测试方法、运行时分析原理、运行时分析分类、Test RealTime 测试工具的使用。

第 4 部分，系统测试：包括功能测试、性能测试，主要讲解系统功能测试所面临的挑战、正则表达式、基于 IBM Rational Robot 的自动化功能测试框架、IBM 性能测试解决方案 Rational Performance Tester。

第 5 部分，验收测试：包括易用性和无障碍测试，主要讲解易用性和无障碍测试的基础理论、软件易用性和无障碍的一系列标准和规范、无障碍测试工具的介绍、易用性和无障碍测试的方法、无障碍测试在 Web 和软件方面的应用。

第 6 部分，案例分析：主要讲解实际案例，要求学生在梳理、总结课程体系中各项知识点和技能的基础上，针对不同的开发阶段，制定相应的测试计划，设计典型测试用例，使用软件测试技术和测试工具，达到测试目标，并进行回归测试，以实现软件测试各单项专业与技能整合运用的目标。

RUP 理论包含了软件开发的所有过程，因此，在学习 RUP 的测试技术之前，应该先对 RUP 的整体理论进行了解，包括 RUP 的特点、原则和概念等，形成对 RUP 的整体理论框架的基本认识，然后是对框架中整个测试体系的分析和理解。建立框架的目的在于，学习完每个测试技术后，能够对所学到的知识有清晰的定位和明确应用的价值，并且能够在学完本教材后，对 RUP 的测试理论有完整的框架型理解，为继续学习 RUP 的其他知识打下基础，将所学到的知识填充进来，最终形成一个完整的 RUP 体系。

学习 RUP 中的几个测试基础时，可以按照工业生产用零件组装模块的惯例，由小到大地进行学习，由函数组装模块，再由模块生成程序，要学习的就是其中能保证生产正确进行的检测方法。可以把开发过程中的函数看作是一个个小零件，由这些零件组成具有特定功能的模块，然后由模块拼接成具有完整功能的系统。生产零件的时候，用单元测试的各种方法来对零件进行检测；把零件组装模块时，用集成测试的方法来检测模块工作是否正常；再用系统测试的方法来检测由模块组合起来的系统是否能正常工作，用性能测试等技术来检查系统性能是否达标，用无障碍测试来检验系统的可用性和关于辅助工具的易用性。

本书适合高等院校相关专业的学生及老师，也适合软件开发、软件测试人员及希望未来从事软件开发和测试的人员阅读。

本书读者应具备：具有计算机的使用经验，学过《软件工程》或者软件测试等基础知识，具有一门高级语言如 C 语言的基本编程基础。

为了帮助读者学习本书，作者除了提供课程网站（www. buu-testing. com）方便读者学习外，还编写了一本《基于 RUP 的软件测试实践题解与上机指导》，提供本书中各章习

题的参考答案以及上机实习指导。该书电子版已放在课程网站提供下载。

关于怎样学习基于 RUP 的软件测试，特提出以下几点看法。

1. 兴趣是最好的老师

如何看待软件测试工作，有人说软件测试很有趣，麻烦堆里有快乐，也有人说它太枯燥。诸多说法表现一个人对这项工作的喜好。但笔者要说的是，如果对软件测试工作没有兴趣和热情很难做到持久。笔者最近参与了一个软件测试项目，在测试团队中，有三位是软件相关专业的本科或研究生在读学生，基础都还不错。但是，只有一位表现最突出，因为他对软件测试抱有浓厚的兴趣，很珍惜这份社会实践的工作机会，做事认真，找出了很多高优先级的 Bug。

另两位同学，在参加项目不到 1 个月后就以各种理由退出了。笔者在与他们的交流中，其中一位同学说测试工作太枯燥，没有挑战性，他更希望做软件开发工作。这位同学喜欢做挑战性的工作无可厚非，但他缺乏对软件测试技术最基本的了解，其实软件测试工作同样具有挑战性。这位同学在 7 天的测试工作中，只找到了 3 个 Bug（正常情况下，一般测试人员每天能找到 5 个 Bug）。因此，在绩效评比中他的成效最低。另一位同学虽然愿意做软件测试，但是他觉得现在的黑盒测试太简单，学习不到测试技术的高级技巧，他更愿意学习白盒测试，能够自己测试软件源代码。而现在他所做的项目没有这部分内容，所以尽管他工作成绩也不错，但是热情不大。

因此，建议同学们在学习或找工作之前，首先需要了解自己对软件测试是否有兴趣或培养起兴趣，是否热爱软件测试工作。只有热爱和专注于某项事业时，才会做出连自己都感到吃惊的成绩来。

2. 巩固测试知识基础

练武术需要先练“蹲马步”，否则直接学习刀枪棍棒等十八般武艺，只能学到几招皮毛。武林高手都是基础牢固，内功深厚。做软件测试也是这个道理。如果认为学完本书，就可以成为测试高手，这是错误的。本书的初衷是为初学者提供一条学习测试实践的捷径，试图将软件测试实践涉及的理论讲述清楚，降低软件测试实践的门槛，引领读者进入基于 RUP 的软件测试实践大门。师傅领进门，修行在个人。要想成为测试高手，还需要个人的努力。个人尽可能多地参加软件测试项目，在实践中学习技能，积累经验，不断分析和总结软件开发过程中可能出错的环节。这样，一名优秀的测试工程师就从软件测试的实践中脱颖而出了。

学习本书需要读者有良好的测试知识基础。RUP 软件测试中主要的测试自动化的质量完全依赖于测试案例和测试数据本身的质量，如果要设计应用于自动化测试的数据，了解划分等价类、确定边界值等测试基础知识是很有必要的。常见一些初学者还没有学会测试的基本概念，就盲目地学习各种大型商业自动化测试软件，结果花了很多时间，只学会了工具的具体操作，而没有实际参与测试的能力。到了实际测试项目中，也无法有效利用工具解决实际测试问题。实际上，测试新手大部分应该从手工功能测试开始起步，只有成为测试高手之后才有能力使用大型自动化测试软件。另外，测试工具的操作是很简

单的技术问题，关键是如何发挥测试工具的作用，这需要了解测试的原理，并通过原理来应用测试工具。

高级的测试人员除了需要精通测试技术，还应掌握行业知识，可以提供行业软件的测试和质量保证方案。对于初学者，要认识到经过不断努力，才可以成为测试行业专家。千里之行，始于足下，目前最重要的是从测试入门知识开始。所以，初学者要老老实实地学习有关测试的基础知识，学习各种测试术语、测试概念、测试分类、测试的流程、测试项目的执行过程等。如果连这些都不懂，今后的职业发展会受到限制。学习测试知识没有捷径，需要从一点一滴学起，日积月累，需要勤奋，需要思考，需要总结提高。

3. 为什么要学习基于 RUP 的软件测试

给大家讲个小故事。有个英国人学煮鸡蛋，开始，他把鸡蛋放到开水里煮时总会炸裂。他为此想了各种方法，并找到了一个解决方案：在鸡蛋上打个孔。但在鸡蛋上打孔带来了另一个问题：孔打小了，鸡蛋还会裂；孔打大了，蛋清会在它凝固以前流出来。于是，这个英国人给一批鸡蛋分别打了各种不同孔径的洞，并记录下每个鸡蛋孔径的大小。通过这一方法，他找到了一个最合适的大小——既避免了炸裂，又保证蛋清不会流出来。这时，虽然煮鸡蛋炸裂的问题解决了，但这个英国人仍然不知道煮多长时间才能把鸡蛋煮熟。为了解决这个问题，他又开始尝试煮不同时间的结果，并从中找出最佳时间。最后他终于摸索到煮鸡蛋的最佳流程和方法。这个小故事对很多中国人来说，可能只是觉得是件可笑的事例，或许认为这个英国人过于迂腐，因为聪明的中国人早就知道把鸡蛋放在冷水中与之一起加热至鸡蛋浮起来就可以了。从煮鸡蛋这样一个小小的事例上，体现了中国人和英国人两种完全不同的思维习惯——中国人凭借自己的经验和聪明直奔结果，而英国人却把每一个过程细化后，框定为任何人任何时候都可操作的流程，然后大家都可按照这个流程执行。

现代产品的开发和生产需要制定一个标准的流程，这个流程并非凭空臆想，它如同英国人煮鸡蛋一样，是经过无数次的探究而得来的。只有制定极为详尽的工业产品生产过程规定，每一步如何做，应该达到何种标准，才能真正地将流程应用于工业生产。因为参与产品生产流程的工人有成百上千，这么多工人只有遵循一套标准的流程，才能生产出许许多多相同的标准产品。

工业产品可以在出厂前设置质量检测来保证质量，但软件毕竟不同于工业产品，它只能在开发过程中监测产品质量。在生产过程中进行质量检测比生产完毕后再进行检测的方法要先进得多，也是保证产品质量的需要，它应该对生产全过程进行追踪。这种方法的思想正是软件过程的质量保证的精髓所在，基于 RUP 的软件测试就是这样的过程质量保证思想。

在欧美国家，如果去应聘软件开发者职位，招聘者就会问："你懂 RUP 吗?"如果说"不懂"，他就会让你回家去学 RUP，然后再来应聘。中国大概还没到这个程度，不过笔者相信很快中国的研发者和经理们也会看到 RUP 的重要性的。

4. 大学生能学好 RUP 理论吗

国内有一种误解，认为 RUP 及其配套软件工具主要用于大型软件开发，学习难度大，不适合初学者。

其实 RUP 并不是高深莫测，只要有学习的基本条件，例如具备计算机、相关软件知识，还要有学习的耐心和毅力，是可以学好 RUP 的。而且想要进入软件开发这个行业，也应该从 RUP 开始学起。从 RUP 中可以学到管理的基本常识、分析设计、体系结构设计、测试，在.NET、J2EE、C++ 或其他平台上的实现，还可以学到进度管理、版本控制、分布式运算等。因为 UML 是软件开发的通用语言，而 RUP 则蕴涵着许多软件开发者应该知道的知识。并且 RUP 是一个稳固的基础，在这个基础之上，可以开发各种各样的软件系统，既可开发小型项目，也可以开发像国防系统、大型银行系统这样庞大的项目。所以，建议有志于从事软件开发和软件测试的，都应该学习 RUP。

有一个学生问笔者，国内最流行是 XP(Extreme Programming，极限编程)方法，RUP 不适合中国国情，为什么不学 XP 反而要学 RUP 呢?

其实这个问题就跟讨论C++ 和 Java 谁好的性质是一样，不能笼统地说哪个好，而且 RUP 不适合的，XP 也未必会适合。刻舟求剑这个成语故事大家知道，如果把 RUP 或 XP 的一整套东西生搬硬套，最后会导致项目失败。正所谓只有最适合的方法，没有最好的方法。

笔者认为 XP 其实就是小型的 RUP。如果要开发一个小型项目，只有很少的团队成员，并且要在比较短的时间内完成，就可以并且应该使用 XP 这种轻量级的方法。这种方法更加灵活，迭代周期更短，但这并不意味着它与 RUP 相对立。实际上随着项目的增大，团队的成长，XP 也可以转变成为 RUP。两者的确有差异，但这种差异也是因为不同项目的需要而造成的。

笔者认为建模是非常重要的，但 XP 却不这么认为。当我们觉得应该建模的时候，XPer 们就已经开始编码了。当然，这是符合 XP 的要求的，的确也获得了很大的成功。不过，这也把 XP 限制在小型项目的范围内。如果在大型项目中这么做，系统很快就会陷入混乱的。当然，这并不是 XP 的错，因为 XP 本来就只是用于小型项目的。

目前，RUP 应用于小型软件企业有一定的难度，而关键在于如何根据项目需要来进行裁减。而且使用 RUP 需要掌握一定的技巧，正如一个高明的铁匠用铁锤可以打出一把好兵器，如果是普通人搞不好还会砸到自己的脚。

5. 加强学习行业知识

建议大家利用一切可以利用的时间学习，多阅读测试书籍，关注和游览测试网站和论坛。要根据自己的知识基础，有选择性地阅读测试书籍。对于初学者应先从基础知识学起。正式出版的书在质量和内容上都有保证，而有些测试网站和论坛的文章良莠不齐，有的只有只言片语，有的还存在一些错误。因此，需要有一定的鉴别能力，否则会被误导，浪费时间。

对于新进入公司的员工，公司一般都要经过短暂的培训，发一些培训材料，这是今后从事工作的最好的第一手材料，针对性和实用性都很强。它是公司测试工作经验的总结，也是今后要用到工作中的一些基本的测试知识和技术的介绍。另外还可以借助测试项目的测试文档学习，包括测试计划、测试用例、测试缺陷数据库，可以先看看以前发现了哪些Bug，这些Bug是怎么被发现的，有什么规律和特征，学习别人怎么写测试缺陷报告。

测试人员除了学习和掌握测试技术外，还需要不断学习行业知识，这是普通测试技术人员与测试行业专家的区别之一。学习什么行业知识呢？根据测试的软件的应用领域决定。例如，如果正在测试的是电信行业的应用软件，那么需要学习电信行业知识，包括术语、业务和行业技术。怎么学习呢？可以与客户交流，与开发人员交流，看专业书和文章。学习行业知识是个不断进步的过程，每个行业都有很系统的知识架构。

6. 测试人员要学会思考

测试是个技术工作，要学会主动思考。测试问题错综复杂，需要自己分析问题的性质，尝试各种解决方法，搜索网上的资料，如果实在解决不了才向测试主管求助。

测试人员如何思考？要根据碰到的问题现象来思考，是属于测试专业知识不足引起的，还是测试用例等测试文档模糊、错误引起的；是个别现象，还是测试项目的其他内容都存在的普遍现象。测试要从模拟用户使用的角度来看，因此要从最终用户的角度来查找问题，分析问题可能导致的严重程度。

在询问最终的解决方法前，必须根据自己的经验尝试了各种解决方法，并且尽量把发现的问题和想法告诉测试主管，证明自己已经主动思考了，只是没有找到更好的解决方法，或者不能确定自己的方法是否可行。

最后我想借用RUP理论创始人之一Ivar Jacobson大师的一句话来送给读者：“我从心底里真诚地告诉中国的开发者：尽快去学习RUP！因为这将是大势所趋。从RUP中，你将可以学到很多很多非常有用的知识。在晚上、在周末、在等女朋友的时候、在喝茶的时候，总之，抽出一切时间来学习RUP。这样的努力必将得到回报。这就是我能给中国的朋友们最好的建议”。

在此，由衷地感谢多年来关心支持本书作者的各位朋友。北京大学杭诚方教授、北京联合大学李启隆教授和毛世春教授等前辈给予了有力的支持和指导，并抽出宝贵的时间审阅此书。郝增茹、崔桓祥、刘文红、蒋雪峰、张鹏、李妍、李明等老师也给予了极大的关心和支持。感谢IBM大学合作部程郁佳、王立女士，IBM信息无障碍研究中心覃玉梅经理等多年来全力支持和帮助作者在计算机教育和计算机普及领域所从事的工作。感谢试用本教材的一些院校，感谢他们提出了一些宝贵的意见。

本书由姚登峰主编，韩玉敏、邱云峰、黄海瑞、李娟参与了部分章节的编写，邱云峰还负责外文校译。另外，黄海瑞和陈林波参与了程序调试工作。由于作者水平和时间有限，难免有缺点和不足，热切期望得到专家和读者的批评指正。

希望本书能促进测试人员基础水平的提高，也为更多的人群加入测试队伍提供辅助向导。如果读者在学习中遇到了难以理解和解决的问题，可以访问本课程网站，也可以直

接和作者进行联系。作者电子邮箱地址为 dfyao@pub.ss.pku.edu.cn。

本书属于原创,但作者的学习成长离不开网络和书籍。在编写此书过程中,作者更是置身于现实的学术氛围,无疑要吸纳和借鉴专家和同行们先进的学术思想、方法,在此深深地感谢他们给予作者的启迪和帮助。

姚登峰

2009 年 6 月于北京西山

目录

第1部分　软件测试基础

第 2 部分　单 元 测 试

第4部分 系统测试

第 5 部分　验 收 测 试

第6部分 案例分析

第 1 部分

软件测试基础

很多人都认为微软是一家软件开发公司，而事实上，我们是一家软件测试公司。

任何技术的第一条法则是，自动化会使高效的行动更加高效。第二条法则是，自动化会使低效的行动更加低效。

——微软公司创始人比尔·盖茨

以上名言，告诉了软件测试的重要性，同时深刻地揭示了自动化技术的真谛。计算机这一门庞大的学科发展至今，它最根本的意义是引入最先进的科学技术，来解决人类手工劳动难以解决的复杂问题，并成为替代人类某些重复性行为模式的最佳工具。基于 RUP 的软件测试技术和自动化测试技术，成为软件测试的最新发展方向，与传统测试技术存在很大的差别。为了帮助读者更好地学习基于 RUP 的软件测试，第 1 章讲述了软件测试基础知识，包括软件测试的起源和发展、测试行业的现状以及优秀测试工程师应该具备的素质；第 2、第 3 章讲解了 RUP 基础以及 RUP 的测试理论；第 4 章介绍了手工测试与自动化测试技术，重点介绍了自动化测试理论和实践。这些都是基于 RUP 的软件测试的重要基础部分。

读者通过这部分的学习，可了解软件测试方面的基础知识，获得基于 RUP 的软件测试理论的基本概念，为下一步的学习打下基础。

第1章 绪论

有关 Bug 的故事

故事发生在 1945 年 9 月的某一天，在一间老式建筑中，突然从窗户外飞进一只飞蛾。美国科学家 Hopper 正埋头在一台名为 Mark Ⅱ的计算机前工作，没有注意到这只即将成为历史事件的飞蛾。这台计算机使用了大量的继电器（电器机械装置，那时还没有使用晶体管）。

突然，Mark Ⅱ死机了。Hopper 试了很多次还是不能启动，Hopper 开始用各种方法查找问题，不知问题出在哪里？最后 Hopper 确定是某个电路板的继电器出错了。Hopper 观察这个出错的继电器，惊奇地发现一只飞蛾躺在里面。Hopper 小心地用镊子将飞蛾夹出来，用透明胶布贴到“时间记事本”中，并诙谐地把飞蛾引起的程序故障称为“Bug”。以后人们便把故障或缺陷称为“Bug”。

Hopper 的事件记事本，连同那只飞蛾，现在都陈列在美国历史博物馆中。这就是软件测试的起源。

开发软件产品和开发其他任何产品一样，都要考虑质量、成本和开发时间等问题，权衡这些因素，找到最佳结合点的难度已经超越解决软件产品的技术问题。

经过多年的研究和实践，人们认识到，仅仅使用几个人在短时间内开发出高质量的产品是不现实的，质量毕竟需要成本和时间来保证。尤其是随着软件规模越来越大，传统的开发模式不适应现代软件开发的要求，而必须以团队合作的形式来组织，团队所有参与者必须统一使用公共的成熟的过程、公共的表达语言以及支持该语言和过程的工具，以保证在可管理的范围下完成软件产品的开发工作。Rational Unified Process（RUP）就是应用于软件开发的一种公共过程，而且已经在很多大型软件开发组织的实践中被证实，可以较好地找到上述问题的最佳结合点，有效地解决软件开发中的各种矛盾，真正保证开发出高质量、低成本的软件产品。

软件测试是软件开发过程中的一项重要工作，是保证软件产品质量、管理与监测的重要手段。在国外大型企业中，软件测试是作为软件开发的重要因素存在，国内的绝大多数软件企业和开发组织者也逐渐认识到软件测试的重要性，增加了软件测试在软件开发过程中的比重。随着软件测试的地位逐步提高，测试理论也在不断发展和完善。RUP 将实践经验与这些成熟的测试理论相融合，形成了 RUP 的测试理论体系。基于这些思想，IBM 公司开发了一系列商业上的测试工具来对测试进行支持，让测试人员从烦琐和重复

的测试活动中解脱出来,通过测试管理和自动化测试等技术来保证软件产品的质量。

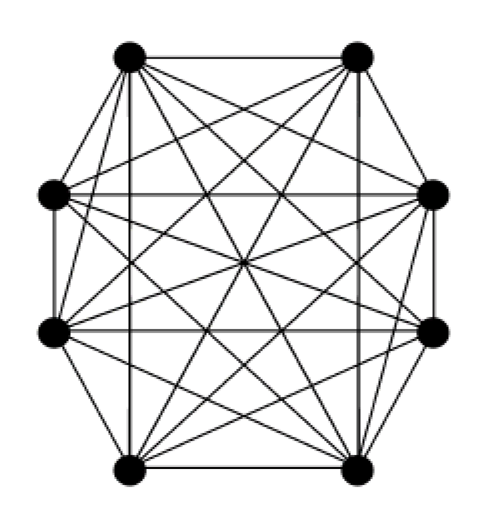
RUP 理论清晰地定义了测试员的不同身份所对应的不同职责。测试员包括测试分析人员、测试管理员、测试设计人员以及测试员。合理地根据需求和工作量安排测试人员,软件测试就能有条不紊地在可管理的状态下完成。但要注意的是,这些职能仅仅在 RUP 中表示不同的个体,而在实际开发过程中,经常由同一个人在同一个或不同项目中同时担任相同或不同的角色。

易用性和无障碍测试近几年进入中国,作为在更高程度上保证软件易用性和通用性的测试,易用性和无障碍测试产生了它自己本身一系列标准和规则,在测试中应用这些标准和规则,对提高软件的易用性和通用性具有非常大的积极意义,也是未来软件测试必须做的重要工作之一。

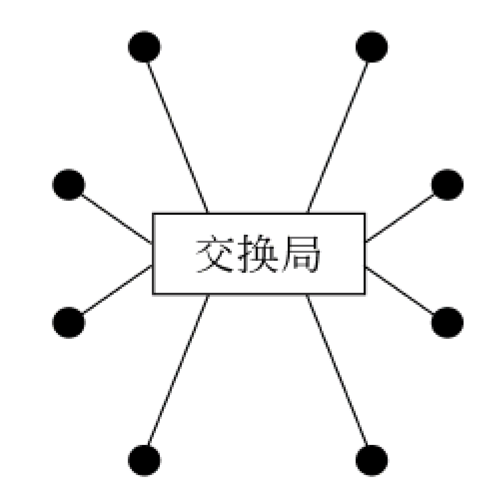
本章简要地介绍 RUP 和软件测试的历史,并进一步说明基于 RUP 的软件测试是怎么产生的。

1.1 引　　言

我们先来回顾一下早期的电话系统。贝尔(Alexander Graham Bell)于 1876 年申请了电话专利。那个时期的电话必须一对一对地卖,用户自己在两个电话之间拉一根线。如果一个用户想和 n 个用户用电话通话,他必须拉 n 根单独的线到每个人的房子里。于是在很短的时间内,城市里到处都是穿过房屋和树木的混乱的电话线。很明显,企图把所有的电话完全互联(见图 1-1(a))是行不通的。



(a) 完全互联的电话系统



(b) 集中交换式的电话系统

图 1-1　电话系统结构模型

贝尔电话公司在 1878 年开办了第一个交换局。公司为每个客户架设一条线。打电话时,客户摇动电话的曲柄使电话公司办公室的铃响起来,操作员听到铃声以后根据要求将呼叫方和被呼叫方用跳线手工连接起来。这种集中交换式的模型如图 1-1(b)所示。很快贝尔系统的交换局就出现在各地。人们又要求能打城市间的长途电话,就出现了二级交换局,以后进一步发展为多个二级交换局,直至现代高科技的网络通信技术。

借用电话通信原理,如果将图 1-1(b)中的电话看成是客户程序,将中心的交换局看成是服务程序,那么图 1-1(b)就是典型的客户机/服务器结构。注意这里客户机和服务器不是指硬件,而是指装了客户程序和服务程序的软件,而且一台计算机可以放多个客户

机和服务器软件。

客户机/服务器结构存在两个明显的优点：

(1) 以集中的方式高效率地管理通信。

(2) 可以共享资源。例如在信息管理系统中，服务器将信息集中起来，任何客户机都可以通过访问服务器而获得所需的信息。

这种客户机/服务器(C/S)体系结构的好处是在保持系统状态不变的情况下，可以快速地对系统做出一些改变，例如创建和修改一些功能，或者引入新的业务规则而不影响当前系统的稳定，也可以随着时间以新的和不可预知的方式来对数据进行整理或修改。这种稳定的体系结构使得许多组织将这种形式应用于他们的开发团队：分析人员和最终领域专家一起将用户的要求转化为软件需求，数据建模人员建立满足用户功能需求的数据模型，应用开发人员则迅速开发出满足需求的新系统。

然而，随着 Web 的出现，软件开发的世界发生了巨大的改变。首先，用户数量向不可控方向发展，传统 C/S 结构的系统中，用户的数目是可以预见并加以控制的，通常是数百或数千；而 Web 上一个系统的用户可能以百万计，并且大多不在系统开发组织的控制之中。其次，在传统的 C/S 系统中，应用和数据在概念上距离相当小，而 Web 上的许多系统都是由数千个可移动的部件组成，其中有使用脚本语言的，有经过编译的，使用的机制与传统的关系型存储大相径庭。再者，在 C/S 系统中，变更虽然是不可避免的，但可以适度地控制；而在 Web 上，变更是持续的，并且在系统体系结构和实现技术的每个层次上都可能会发生变更。最后，在 C/S 系统中，参与系统开发的人员相对较少，而在 Web 上除了传统的软件开发团队以外，还有许多新的参与者，从内容创建人员、信息架构构造人员到网络设计人员，大家共同合作进行软件的开发。

C/S 系统在实现上是相对同构的，因为 C/S 系统中服务器和客户端系统之间是完全相同的，数据库表结构也是相同的，表现为具有相同的硬件平台、系统软件的环境。而 Web 的出现改变了这点。Web 系统可能在服务器端使用C++ 或 Java，或者少量的脚本语言，在客户端使用传统的语言如 VB 和 Java，有时也会使用诸如 XML 这样的语言。除了面对这些编程语言，开发人员还必须具备处理庞杂问题的技术能力，如 Microsoft 的 WinDNA、Sun 的 EJB 等，这些技术提供给开发人员的编程模型也各不相同。这样 Web 服务器和客户端系统之间存在一定的差异性，数据库表结构也可能不相同。

为了解决这种差异性，开发团队成员能够以一种通用的方式沟通是至关重要的，不同的参与者对系统的设计和实现有不同的看法，只有使用通用的语言表达，才可能统一开发团队的活动。

除了要解决通用语言的问题，软件开发中还有几个问题决定其开发产品的成败。第一点是软件的创新思路，当前功能相似的软件很多，但是好的创新思路却能让软件产品脱颖而出。第二点是如何将创新思路转化为适应市场需求且高质量、低成本的产品。当今社会技术的更新日新月异，因产品落后于时代而失败的案例数不胜数，但如果过于强调开发速度而忽略产品的质量，同样也会让软件产品失去竞争力。同样，如果不注意成本控制，造成过多的无效支出，会提高软件产品的价格，也将降低其产品的竞争力。如何在这几者之间取得平衡，正是当今软件工程所要做的工作。此外在软件开发中还有许多问题，

如大规模、异地分布、并发和团队开发等。要成功解决这些问题，就必须以团队活动形式来进行软件开发，团队中从事开发和部署的不同参与者统一使用公共的过程，还有前面提到的通用语言以及支持该语言和过程的工具。

James Rumbaugh、Grady Booch 和 Ivar Jacobson 三位科学家提出了 RUP 理论和 UML（Unified Modelling Language，统一建模语言），具有划时代意义。RUP 就是解决上述问题的一种公共过程，它定义了一系列的开发理论和管理方法，可以适应大多数的软件开发过程，并且能够根据企业管理方法和软件开发的特殊需要来对开发过程模型进行更改，以达到适应企业本身和软件开发的需要。能够有效管理软件开发，通过设计一整套完整的灵活的软件开发过程，提高软件开发效率。作为开发团队的公共语言，UML 是一种直观的、明确的图形化语言，它包含了当前主流建模技术的概念，是目前软件建模语言的标准。RUP 和 UML 作为孪生兄弟，在过去的几年中，RUP 和 UML 已经在很多软件开发组织的实验中被证实，可以有效地解决上述软件开发中的问题。越来越多的软件开发组织开始使用 RUP 来管理软件开发，利用它来提升企业本身的竞争力。

1.2 错误是不可避免的

信息技术的飞速发展，使软件产品渗透并应用到千家万户以及社会的各行各业的各个领域，软件产品的质量自然成为人们共同关注的焦点。不论软件的生产者还是软件的使用者，都生存在竞争的环境中，软件开发商为了占有市场，必须把产品质量作为企业追求的重要目标，以免在激烈的竞争中被淘汰出局；用户为了保证自己业务顺利完成，当然希望选用优质的软件。质量不佳的软件产品不仅会使开发商的维护费用和用户的使用成本大幅增加，还可能产生其他的风险责任，造成公司信誉下降和其他一系列严重后果。在一些重要领域，如民航订票系统、银行结算系统、证券交易系统、自动飞行控制软件、军事防御和核电站安全控制系统等，如果使用质量有问题的软件，还可能造成灾难性的损失。

软件危机曾经是软件界甚至整个计算机界最热门的话题。为了解决这场危机，软件从业人员、专家和学者做出了大量的努力。现在人们已经逐步认识到所谓的软件危机实际上仅是一种状况，那就是软件中有错误，正是这些错误导致了软件开发在成本、进度和质量上的失控。有错是软件的属性，而且是无法改变的，因为软件是由人来完成的，所有由人做的工作都不会是完美无缺的。

给软件带来错误的原因很多，具体地说，主要有如下几点：

（1）交流不够、交流误解或根本不进行交流

在软件的需求分析未确定或需求发生变更的情况下，如果没有及时交流，开发出来的程序即使功能完整，也属于错误的开发。现在的大型系统通常由几十至几百几千人共同开发，中间的哪一个环节交流不够都有可能造成整个系统开发的延误。

（2）软件复杂性

图形用户界面（GUI）、客户机/服务器结构、分布式应用、数据通信、超大型关系型数据库以及庞大的系统规模，使得软件及系统的复杂性呈指数增长，没有现代软件开发经验

的人很难理解它。

（3）程序设计错误

像所有的人一样，程序员也会出错。这包括语句错误和逻辑错误，语言设计工具能够检测出一部分语句错误，但是对于部分语句错误和几乎所有逻辑错误都是无法被语言设计工具检查出来的，这些错误就会一直存在于程序中。

（4）需求变化

需求变化的影响是多方面的，客户可能不了解需求变化所带来的影响，也可能知道但又不得不那么做。需求变化的后果可能是造成系统重新设计，设计人员日程重新安排，已经完成的工作可能要部分重做或者完全抛弃，对其他项目产生影响，硬件需求可能要因此改变等。如果有许多小的改变或者一次变化，项目各部分之间已知或未知的依赖性可能会相互影响而导致更多问题的出现，需求改变带来的复杂性可能导致错误，还可能影响工程参与者的积极性。

（5）时间压力

软件项目的日程表很难做到准确，很多时候需要预计和猜测。当最终期限迫近和关键时刻到来之际，必须加紧工作，提高开发速度，在没有完整的除错方案或没有时间进行完整的除错检查的情况下，错误也就跟着来了。

（6）开发人员的自负

他们经常说“没问题！”、“这事情很容易！”、“几个小时我就能拿出来！”等之类太多不切实际的“没问题”，结果只能是引入错误。

（7）代码文档贫乏

贫乏或者较差的文档使得代码维护和修改变得异常艰辛，其结果是带来许多错误。现在许多机构不鼓励程序员为代码编写文档，也不鼓励程序员将代码写得清晰和容易理解，相反他们认为少写文档可以更快地进行编码，无法理解的代码更有利于加强工作的保密性。这种做法是愚昧的，也是带来软件错误的原因之一。

（8）软件开发工具

可视化工具、类库、编译器、脚本工具等常常会将自身的错误带到软件中。就像我们所知道的，没有良好的工程化作为基础，使用面向对象的技术只会使项目变得更复杂。为了解决这些问题，软件界做出了各种各样的努力。

造成错误的原因有很多，关键在于如何去避免错误的产生和消除已经产生的错误，使程序中的错误密度尽可能地降低。人们曾经认为更好的程序语言可以摆脱这些困扰，这样推动了程序设计语言的发展。为了使程序更易于理解，开发了结构化程序设计语言，如PL/1、PASCAL等；为了解决实时多任务需求，开发了结构化多任务程序设计语言，如Modula、Ada等；为了提高重用性，开发了面向对象的程序设计语言，如Simlasa等；为了避免产生不正确的需求理解，开发了形式化描述语言，如HAL/S等，这使得建立基于自然语言的描述成为可能；为了支持大型数据库应用，开发了可视化工具，如Visual Studio、Power Builder等。程序语言对提高软件生产效率起到了一定的积极作用，但它对整个软件质量尤其是可靠性的影响就显得微弱了。

可能是因为程序语言严格的语法和语义规则，人们曾经试图用形式化证明方法来证

明程序的正确性。数学家对形式化证明方法最有兴趣，将程序当作数学对象来看待，从数学意义上证明程序的正确的确是可能的。从理论上来看非常吸引人，但实际价值却非常有限，因为形式化证明方法只有在代码写出来之后才能使用，这时再来证明显然太迟了，而且对于大的程序证明起来也非常困难。受到其他行业项目工程化的影响和启发，软件被视为一项工程，软件工程学被提了出来，企业开始以工程化的方法来进行规划和管理软件的开发。

在可以借助许多新的技术和工具进行软件开发的今天，软件开发过程的成熟性问题开始引起人们的重视。经过研究，人们发现影响软件开发的主要症结在于管理，因此人们将目标转向改善管理，一些以改进软件开发过程管理为目标的活动已经展示出积极的效果。

但是不论采用什么技术和什么方法，软件中仍然会有错。采用新的语言、先进的开发方式和完善的开发过程，可以减少错误的引入，但是不可能完全杜绝软件中的错误，这些引入的错误需要通过测试来找出，软件中的错误密度也需要通过测试来进行评估。测试是所有工程学科的基本组成元素，是软件开发的重要部分。自有程序设计的那天开始，测试就一直伴随着程序设计的产生和发展。有关统计表明，在典型的软件开发项目中，软件测试工作量往往占软件开发总工作量的 40%以上。而在软件开发的总成本中，用在测试上的开销要占 30%到 50%。如果把维护阶段也考虑在内，包括整个软件生存期时，测试的成本比例也许会有所降低，但实际上维护工作相当于二次开发，乃至多次开发，其中必定还包含许多测试工作。因此，测试对于软件生产来说是必需的，问题是我们应该思考“采用什么方法？如何安排测试？”

1.3 软件测试历史

软件测试是伴随着软件的产生而产生的。在早期的软件开发过程中，软件规模很小，复杂程度低，软件开发的过程相当随意，混乱无序，测试的含义也比较狭窄。开发人员将测试等同于“调试”，目的是纠正软件中已经知道的故障，常常由开发人员自己完成这部分的工作，对测试的投入极少，测试介入也晚，常常是等到形成代码，产品已经基本完成时才进行测试。

直到 1957 年，软件测试才开始与调试区别开来，成为一种发现软件缺陷的活动。由于一直存在着首先确信产品能工作后方能检测的思想，测试工作仍然滞后于开发活动。到了 20 世纪 70 年代，尽管对“软件工程”的真正含义还缺乏认识，但这一词条已经频繁出现，一些软件测试的探索者建议在软件生命周期的开始阶段就根据需求制订测试计划，这时也涌现出一批软件测试的宗师，Bill Hetzel 博士就是其中的领导者。1972 年软件测试领域的先驱 Bill Hetzel 博士在美国的北卡罗来纳大学组织了历史上第一次正式的关于软件测试的会议。1973 年，他首先给软件测试提出这样的定义：“就是建立一种信心，认为程序能够按预期的设想运行。”后来在 1983 年，他又将定义修订为：“评价一个程序和系统的特性或能力，并确定它是否达到预期的结果。软件测试是以此为目的的任何

行为。”

1975 年，John Good Enough 和 Susan Gerhart 在 IEEE 上发表了《测试数据选择的原理》的文章，软件测试才被确定为一种研究方向。而 1979 年，Glenford Myers 的《软件测试艺术》(The Art of Software Testing)可算是软件测试领域的第一本重要的专著，Myers 作为当时最好的软件测试人员之一，他定义了：“测试是为发现错误而执行的一个程序或者系统的过程”。Myers 以及他的同事在 20 世纪 70 年代的工作是测试过程发展的里程碑。

然而，对 Glenford Myers 先生“测试的目的是证伪”这一概念的理解也不能过于片面。在很多软件工程学、软件测试方面的书籍中都提到一个概念：“测试的目的是寻找错误，并且是尽最大可能找出最多的错误”。这很容易让人们认为测试人员就是“挑毛病”的，而由此带来诸多问题。大家熟悉的 Ron Patton 在《软件测试》(中文版由机械工业出版社出版，此书是目前国内测试新手入门的经典教材)一书的第 10 页，有一个明确而简洁的定义：“软件测试人员的目标是找到软件缺陷，尽可能早一些，并确保其得以修复。”这样的定义具有一定的片面性，带来的结果是：

① 若测试人员以发现缺陷为唯一目标，而很少去关注系统对需求的实现，测试活动往往会存在一定的随意性和盲目性；

② 若有些软件企业接受了这样的方法，以 Bug 数量来做为考核测试人员业绩的唯一指标，也不太科学。

到了 20 世纪 80 年代初期，IT 行业包括软件业进入大发展时期，软件趋向大型化、高复杂度，软件的质量越来越重要，“质量”的号角开始吹响。软件测试定义发生了根本的变化，测试不单纯是一个发现错误的过程，而且包含软件质量评价的内容。这表明人们对软件测试的认识更趋于科学。软件开发人员和测试人员开始坐在一起探讨软件工程和测试问题，这个时候，一些软件测试的基础理论和实用技术开始形成，人们开始为软件开发设计了各种流程和管理方法，软件开发的方式也逐渐由混乱无序的开发过程过渡到结构化的开发过程，以结构化分析与设计、结构化评审、结构化程序设计以及结构化测试为特征。他们还制定了软件测试行业标准，包括 IEEE(Institute of Electrical and Electronic Engineers)标准、美国 ANSI(American National Standard Institute)标准以及 ISO(International Standard Organization)国际标准。

1983 年 IEEE 提出的软件工程术语中给软件测试下的定义是：“使用人工或自动的手段来运行或测定某个软件系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别”。这个定义明确指出：软件测试的目的是为了检验软件系统是否满足需求。它再也不是一个一次性的而且只是开发后期的活动，而是与整个开发流程融合为一体。软件测试已成为一个专业，需要运用专门的方法和手段，需要专门人才和专家来承担。

1983 年，Bill Hetzel 在《软件测试完全指南》一书中指出：“测试是以评价一个程序或系统属性为目标的任何一种活动，测试是对软件质量的度量”。Myers 和 Hetzel 的定义至今仍被引用。到了 2002 年，Rick 和 Stefan 在《系统的软件测试》(Systematic Software Testing)一书中对软件测试做了进一步定义：“测试是为了度量和提高被测软件的质量，

对测试软件进行工程设计、实施和维护的整个生命周期过程”。这些经典论著对软件测试研究的理论化和体系化产生了巨大的影响。

随着软件产业界对软件过程的不断研究，美国工业界和政府部门开始认识到，软件过程能力的不断改进才是增进软件开发组织的开发能力，提高软件质量的第一要素。在这种背景下，由美国卡内基-梅隆大学软件工程研究所(SEI)研制并推出了软件能力成熟度模型 SW-CMM，CMM 逐渐成为了评估软件开发过程的管理以及工程能力的标准。但是令人遗憾的是，CMM 没有充分的定义软件测试，没有提及测试成熟度的概念，没有对测试过程改进进行充分说明，在 KPA 中没有定义测试问题，与质量相关的测试问题如可测性、充分测试标准、测试计划等方面也没有满意的阐述。为此，许多研究机构和测试服务机构从不同角度出发，提出有关软件测试方面的能力成熟度模型，作为 SEI-CMM 的有效补充，比较有代表性的包括 Burnstein 博士提出了测试成熟度模型(TMM)，依据 CMM 的框架提出测试的 5 个不同级别，关注于测试的成熟度模型。它描述了测试过程是项目测试部分得到良好计划和控制的基础。

1998 年 James Rumbaugh、Grady Booch 以及 Ivar Jacobson 三位科学家提出了 UML 语言和 RUP 理论，还创建了 Rational 公司。基于 RUP 的软件开发过程越来越受到广泛的关注，不断地增强和扩展了 UML 的设计和开发过程。基于 RUP 的软件测试过程却被人们忽略，其实三位科学家同时也首次提出了基于 RUP 的软件测试，将软件测试理论进一步深化，提升到一个新的高度。在三位科学家看来，软件测试是软件开发中极为重要的过程，在很多软件开发组织，测试在整个软件开发过程中所占的比例约为 40%。测试的发展将直接关系到软件产品的质量，同时，这也是 UML 软件开发过程不断完善的有效保证。因此在 RUP 的六大经验基础上，还提出基于 RUP 的软件测试理念，核心是尽早测试、连续测试、自动化测试，并在此基础上提供了完整的软件测试流程和一整套软件自动化测试工具，使我们最终能够做到：一个测试团队，基于一套完整的软件测试流程，使用一套完整的自动化软件测试工具，完成全方位的软件质量验证。所以，基于 RUP 的软件测试正成为国内外研究的一个热点。

软件测试的一个发展趋势是与其他学科融合交叉，例如 1998 年美国北卡罗来纳州州立大学 Ron Mace 教授提出了通用设计的重要理念，可用性测试的概念也随之诞生，可用性测试横跨工业设计、心理学、行为学、人机工程学、美学等学科，主要用于验收测试阶段。可用性测试是用来了解用户对这个新产品的反应，同时发现一些项目小组所忽略的使用上的问题。当时可用性测试包含了无障碍测试的雏形，此后来自 IBM 公司的研究员 Jim Thater 博士于 2003 年在他的书《Constructing Accessible Web Sites》中首次提出了无障碍测试的概念，并将无障碍测试从可用性测试学科中分离出来，发扬光大。在此基础上，IBM 公司形成了完整的无障碍测试流程、企业内部无障碍规范、一整套无障碍测试自动化工具、无障碍测试理念等，从而使基于 RUP 的无障碍测试理论更为成熟，更有助于保证 IBM 公司软件产品的质量，为企业自身及其客户带来效益和价值。

近 20 年来，随着计算机和软件技术的飞速发展，软件测试技术研究也取得了很大的突破，但是其发展速度仍落后于软件开发技术的发展速度，使得软件测试在今天面临着很大的挑战。可以预见，在未来的几年内，软件测试理论还会得到不断更新和更快的发展，

将会更好地指导实践，保证软件质量。

1.4 软件测试模型的演变

虽然大多数人都认同模型的重要性，但在开发周期中，测试模型并没有受到应有的关注，这里简单介绍一下测试模型的演变。软件测试模型与软件测试标准的研究也随着软件工程的发展而越来越深入，在20世纪80年代后期，Paul Rook提出了著名的软件测试的V模型，旨在改进软件开发的效率和效果。V模型反映出了测试活动与分析设计活动的关系。图1-2从左到右描述了基本的开发过程和测试行为，非常明确地标注了测试过程中存在的不同类型的测试，并且清楚地描述了这些测试阶段和开发过程中各阶段的对应关系。

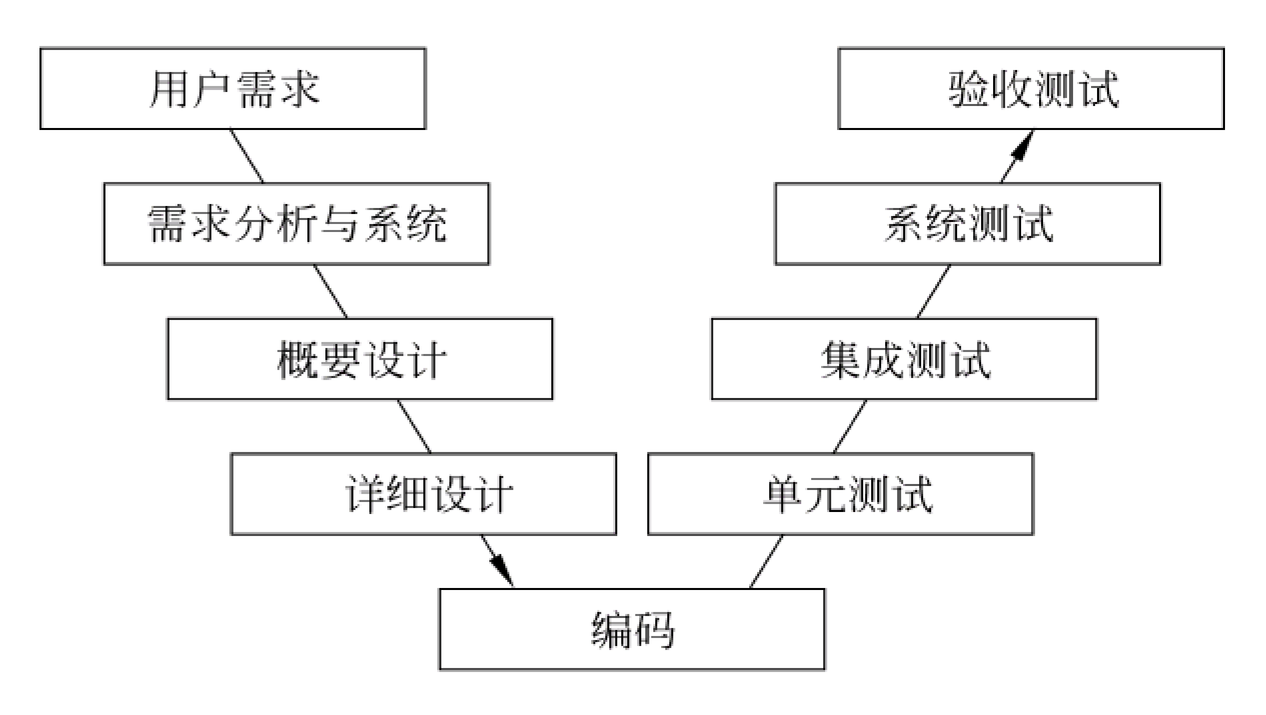


图1-2 开发过程和测试行为的关系

国内大多数企业采用V模型作为测试的标准模型，来规划和设计软件测试流程，指导日常的测试工作。V模型指出，单元和集成测试应检测程序的执行是否满足软件设计的要求；系统测试应检测系统功能、性能的质量特性是否达到系统要求的指标；验收测试确定软件的实现是否满足用户需要或合同的要求。但V模型存在一定的局限性，它仅仅把测试作为在编码之后的一个阶段，是针对程序进行的寻找错误的活动，而忽视了测试活动对需求分析、系统设计等活动的验证和确认的功能。

Evolutif公司针对V模型的缺陷，提出了W模型的概念，W模型增加了软件各开发阶段中应同步进行的验证和确认活动。如图1-3所示，W模型由两个V字形模型组成，分别代表测试与开发过程，图中明确表示出了测试与开发的并行关系。

W模型强调：测试伴随着整个软件开发周期，而且测试的对象不仅仅是程序，需求、设计等同样要测试，也就是说，测试与开发是同步进行的。W模型有利于尽早地全面的发现问题。例如，需求分析完成后，测试人员就应该参与到对需求的验证和确认活动中，以尽早地找出缺陷所在。同时，对需求的测试也有利于及时了解项目难度和测试风险，及早制定应对措施，这将大大减少总体测试时间，加快项目进度。

但W模型也存在局限性。在W模型中，需求、设计、编码等活动被视为串行的，同时，测试和开发活动也保持着一种线性的前后关系，上一阶段完全结束，才可正式开始下

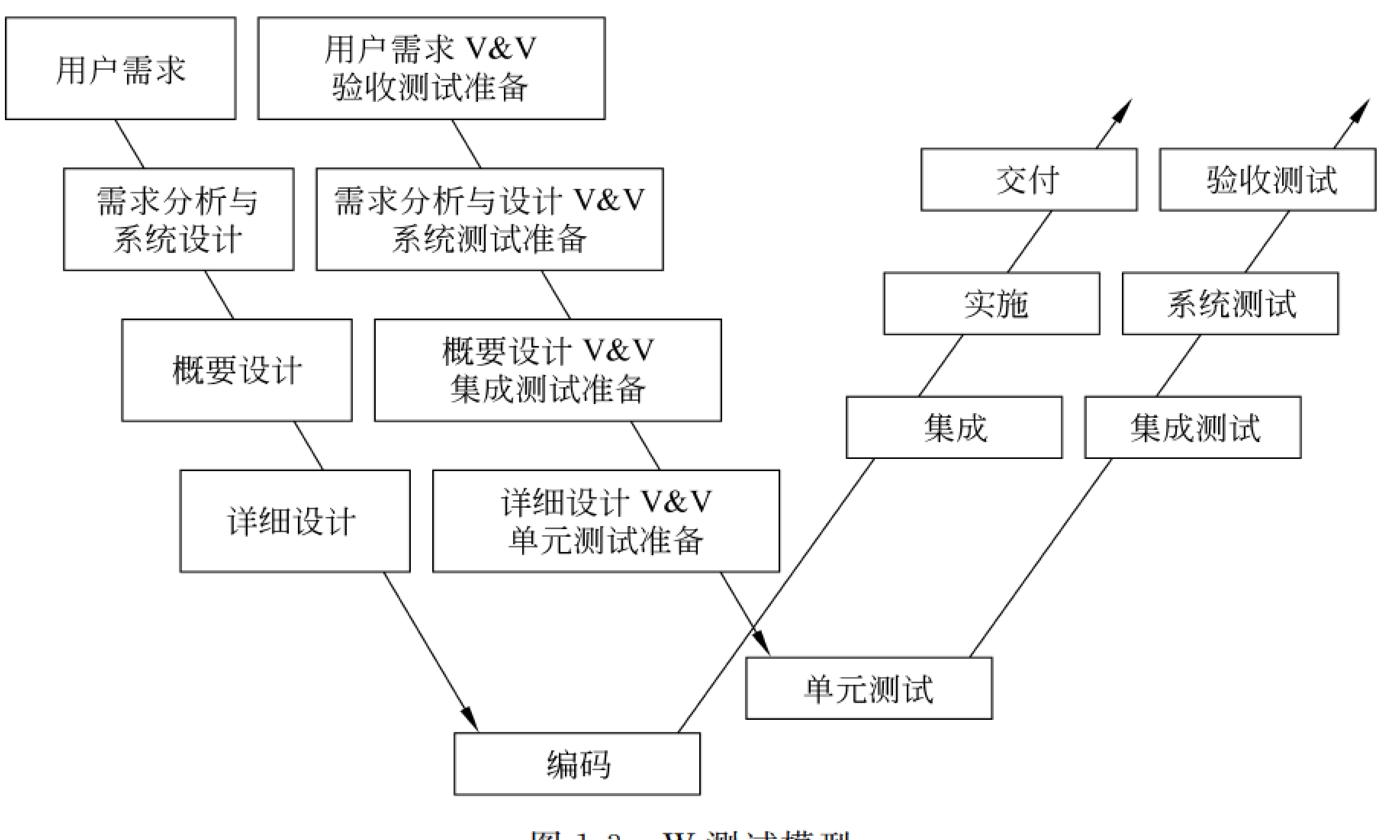


图 1-3　W 测试模型

一个阶段工作。这样就无法支持迭代的开发模型。对于当前软件开发复杂多变的情况，W 模型并不能解除测试管理面临的困惑。

V 模型和 W 模型均存在一些不妥之处。如前所述，它们都把软件的开发视为需求、设计、编码等一系列串行的活动，而事实上，这些活动在大部分时间内是可以交叉进行的，所以，相应的测试之间也不存在严格的次序关系。同时，各层次的测试(单元测试、集成测试、系统测试等)也存在反复触发、迭代的关系。

为了解决以上问题，有专家提出了 H 模型。它将测试活动完全独立出来，形成了一个完全独立的流程，将测试准备活动和测试执行活动清晰地体现出来，如图 1-4 所示。

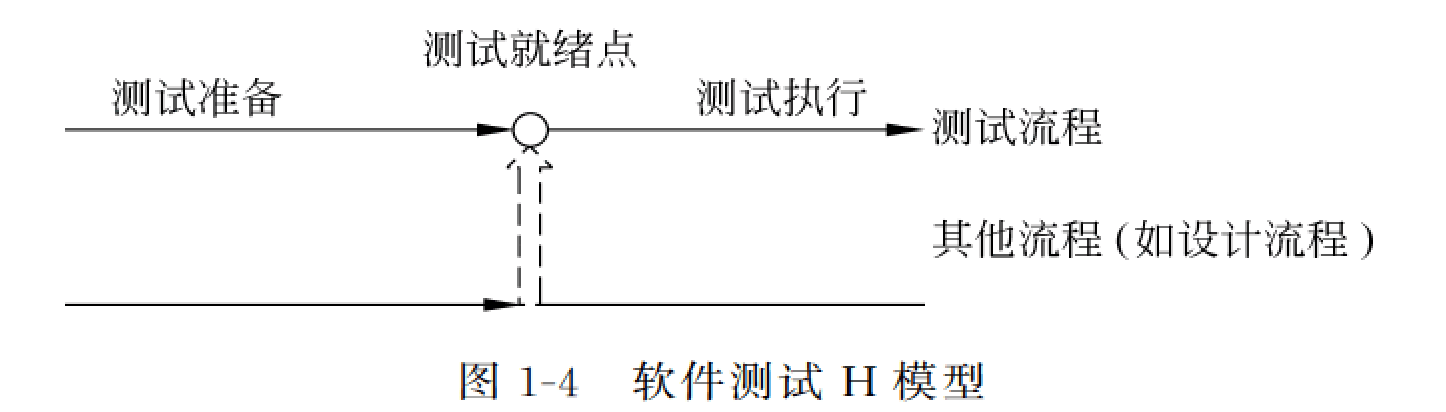


图 1-4　软件测试 H 模型

这个示意图仅仅演示了在整个生产周期中某个层次上的一次测试“微循环”。图中标注的其他流程可以是任意的开发流程。例如，设计流程或编码流程。也就是说，只要测试条件成熟了，测试准备活动完成了，测试执行活动就可以进行了，或者说需要进行了。

H 模型揭示了一个原理：软件测试是一个独立的流程，贯穿产品整个生命周期，与其他流程并发地进行。H 模型指出软件测试要尽早准备，尽早执行。不同的测试活动可以是按照某个次序先后进行的，但也可能是反复的，只要某个测试达到准备就绪点，测试执行活动就可以开展。

以上模型跟 RUP 的关系很大，RUP 提出的“尽早测试”和“全过程测试”思想就是从 W 模型中抽象出来的理念，W 模型认为测试并不是在代码编写完成之后才开展的工作，

测试与开发是两个相互依存的并行的过程，测试活动在开发活动的前期已经开展。而且W模型形象地表明了软件开发与软件测试的紧密结合，这就说明软件开发和测试过程会彼此影响，这就要求测试人员对开发和测试的全过程进行充分的关注。

软件测试与软件开发是紧密结合的，但并不代表测试是依附于开发的一个过程，测试活动是独立的。这正是H模型所主导的思想。“独立的、迭代的测试”着重强调了测试的就绪点，也就是说，只要测试条件成熟，测试准备活动完成，测试的执行活动就可以开展。这个思想被RUP采用作为实践经验之一。

除上述几种常见模型外，还有X模型、前置测试模型等。X模型提出针对单独的程序片段进行相互分离的编码和测试，此后通过频繁的交接，通过集成最终合成为可执行的程序。前置测试模型体现了开发与测试的结合，要求对每一个交付内容进行测试。随着研究的深入，业界还新出来几种模型，例如基于敏捷开发的X模型、非间断软件回归测试模型、基于构件技术的软件测试模型等，这些模型都针对其他模型的缺点提出了一些修正意见，但本身也可能存在一些不周全的地方，没有一种测试模型能够完全地适用所有的软件开发过程。一般要谨慎分析，正确选取测试模型。

1.5 软件测试类型

一次文学考试中有一道题，问高尔基是哪国人。一考生乐极而吟：“尔基啊尔基，你若不姓高，我怎知你是中国人。”这是一种典型的瞎猜法。如果这种方法用于软件测试，人累死也测不出任何结果。

软件测试是一门科学，需要科学的方法和态度。

不论是对软件的模块还是整个系统，总有共同的内容要测试、如正确性测试、容错性测试、性能与效率测试、易用性测试、文档测试等。白盒测试是指开发人员从程序内部对上述内容进行测试，而黑盒测试是指独立的测试人员从程序外部对上述内容进行测试。很多软件工程教材讲述了各种各样的测试方法并列举了不少示例。本节主要介绍常见的软件测试类型。

1. 回归测试（Regression Test）

回归测试是指修改了旧代码后，重新进行测试，以确认修改没有引入新的错误或导致其他代码产生错误。回归测试的目的在于验证以前出现过，又已经修复好的缺陷不再重新出现，一般对某已知修正的缺陷再次围绕它原来出现时的步骤重新测试。因为为了修正某缺陷时必须更改源代码，因而就有可能影响这部分源代码所控制的功能。所以在验证修好的缺陷时不仅要服从缺陷原来出现时的步骤重新测试，而且还要测试有可能受影响的所有功能。因此RUP鼓励对所有回归测试用例进行自动化测试。

2. 构建时确认测试（Build Verification Test，BVT）

BVT是在所有开发工程师都已经检入自己的代码，项目组编译生成当天的版本之后进行，主要目的是验证最新生成的软件版本在功能上是否完整，主要的软件特性是否正

确。如无大的问题，就可以进行相应的功能测试。BVT 优点是时间短，验证了软件的基本功能。缺点是该种测试的覆盖率很低。因为运行时间短，不可能把所有的情况都测试到。

3. 基于用户实际应用场景的测试（Scenario Test）

在做 BVT、功能测试的时候，可能测试主要集中在某个模块或比较分离的功能上。当用户来使用这个应用程序的时候，各个模块是作为一个整体来使用的，那么在做测试的时候，就需要模仿用户这样一个真实的使用环境，即用户会有哪些用法，会用这个应用程序做哪些事情，操作会是一个怎样的流程。加了这些测试用例后，再与 BVT、功能测试配合，就能使软件整体都能符合用户使用的要求。Scenario Test 的优点是关注了用户的需求，缺点是有时候难以真正模仿用户真实的使用情况。

4. 冒烟测试（Smoke Test）

在测试中发现问题，找到了一个 Bug，然后开发人员会来修复这个 Bug。这时想知道这次修复是否真的解决了程序的 Bug，或者是否会对其他模块造成影响，就需要针对此问题进行专门测试，这个过程就被称为 Smoke Test。在很多情况下，做 Smoke Test 是开发人员在试图解决一个问题的时候，造成了其他功能模块一系列的连锁反应，原因可能是只集中考虑了一开始的那个问题，而忽略其他问题，这就可能引起了新的 Bug。Smoke Test 优点是节省测试时间，防止 Build 失败。缺点是覆盖率还是比较低。

此外，兼容性测试（Application Compatibility Test）主要目的是为了兼容第三方软件，确保第三方软件能正常运行，用户不受影响。无障碍测试（Accessibility Test）是确保软件对于某些残疾人士也能正常地使用。其他测试还有功能测试（Functional Test）、安全性测试（Security Test）、压力测试（Stress Test）、性能测试（Performance Test）和安装升级测试（Setup/Upgrade Test）等。

微软的软件测试工作

（1）基本情况

测试在微软公司是一项非常重要的工作，微软公司在此方面的投入也是巨大的。微软对测试的重视表现在工程开发队伍的人员构成上，微软的项目经理、软件开发人员和测试人员，三方面人员的比例是 1∶3∶3 或 1∶4∶4，可以看出开发人员与测试人员的比例基本是 1∶1。对于测试的重视还表现在最后产品要发布的时候，此产品的所有相关部门都必须签字，而且测试人员具有绝对的否决权。

测试人员中分成两种职位，Software Development Engineer in Test（测试组的软件开发工程师）实际上还是属于开发人员，他们具备编写代码的能力和开发工具软件的经验，侧重于开发自动化测试工具和测试脚本，实现测试的自动化。Software Test Engineer（软件测试工程师）具体负责测试软件产品，主要完成一些手工测试以及安装配置测试。

（2）测试计划

测试计划是测试人员管理测试项目，在软件中寻找 Bug 的一种有效的工具。测试计

划主要有两个作用，一是评判团队的测试覆盖率以及效率，让测试工作有条不紊地展开。二是有利于与项目经理、开发人员进行沟通。有了测试计划之后，他们就能够知道是如何开展测试工作的，他们也会从中提出很多有益的意见，确保测试工作顺利进行。总之，有了测试计划可以更好地完成测试工作，确保用户的满意度。

(3) 测试用例开发

一个好的测试用例就是有一个合理的概率来找到 Bug，不要冗余，要有针对性，一个测试只针对一件事情。特别是功能测试的时候，如果一个测试测了两项功能，那么如果测试结果失败的话，就不知道到底是哪项功能出了问题。

测试用例开发中主要使用的技术有等价类划分，边界值的分析，Error Guessing Testing。

① 等价类划分是根据输入输出条件，以及自身的一些特性分成两个或更多个子集，来减少所需要测试的用例个数，并且能用很少的测试用例来覆盖很多的情况，减少测试用例的冗余度。在等价类划分中，最基本的划分是一个为合法的类，一个为不合法的类。

② 边界值的分析是利用了一个规律，即程序最容易发生错误的地方就是在边界值的附近，它取决于变量的类型，以及变量的取值范围。一般对于有 n 个变量时，会有 $6n+1$ 个测试用例，取值分别是 min－1，min，min＋1，normal，max－1，max，max＋1 的组合。边界值的分析的缺点，是对逻辑变量和布尔型变量不起作用，还有可能会忽略掉某些输入的组合。

③ Error Guessing Testing 完全靠的是经验，所设计的测试用例就是常说的猜测。感觉到软件在某个地方可能出错，就去设计相应的测试用例，这主要是靠实际工作中所积累的经验和知识。其优点是速度快，只要想得到，就能很快设计出测试用例。缺点就是没有系统性，无法知道覆盖率会有多少，很可能会遗漏一些测试领域。

实际上微软是采用一些专门的软件或工具负责测试用例的管理，有一些测试信息可以被记录下来，例如测试用例的简单描述，在哪些平台执行，是手工测试还是自动测试，运行的频率是每天运行一次，还是每周运行一次。此外还有清晰的测试通过或失败的标准，以及详细记录测试的每个步骤。

(4) Bug 跟踪过程

在软件开发项目中，测试人员的一项最重要使命就是对所有已知 Bug 进行有效的跟踪和管理，保证产品中出现的所有问题都可以得到有效的解决。一般项目组发现、定位、处理和最终解决一个 Bug 的过程包括 Bug 报告、Bug 评估和分配、Bug 处理、Bug 关闭等如下四个阶段。

① 测试工程师在测试过程中发现新的 Bug 后，应向项目组报告该 Bug 的位置、表现、当前状态等信息，项目组在 Bug 数据库中添加该 Bug 的记录。

② 开发经理对已发现的 Bug 进行集中讨论，根据 Bug 对软件产品的影响来评估 Bug 的优先级，制定 Bug 的修正策略。按照 Bug 的优先级顺序和开发人员的工作安排，开发经理将所有需要立即处理的 Bug 分配给相应的开发工程师。

③ 开发工程师根据安排对特定的 Bug 进行处理，找出代码中的错误原因，修改代码，重新生成产品版本。

④ 开发工程师处理了 Bug 之后,测试人员需要对处理后的结果进行验证,经过验证确认已正确处理的 Bug 被标记为关闭状态。测试工程师既需要验证 Bug 是否已经被修正,也需要确定开发人员有没有在修改代码的同时引入新的 Bug。

(5) Bug 的不同处理方式

在某些情况下,Bug 已处理并不意味着 Bug 已经被修正。开发工程师可以推迟 Bug 的修正时间,也可以在分析之后告知测试工程师这实际上不是一个真正的 Bug。也就是说,某特定的 Bug 经开发工程师处理之后,该 Bug 可能包括以下几种状态。

① 已修正。开发工程师已经修正了相应的程序代码,该 Bug 不会出现了。

② 可推迟。该 Bug 的重要程度较低,不会影响当前应提交版本的主要功能,可安排在下一版本中再行处理。

③ 设计问题。该 Bug 与程序实现无关,其所表现出来的行为完全符合设计要求,对此应提交给程序经理处理。

④ 无须修正。该 Bug 的重要程度非常低,根本不会影响程序的功能,项目组没有必要在这些 Bug 上浪费时间。

1.6 软件测试工具的发展

进入 20 世纪 90 年代,软件的规模变得异常庞大。在一些大型软件开发过程中,测试活动需要花费大量的时间和成本,而当时测试的手段几乎完全都是手工测试,测试的效率非常低,并且随着软件复杂度的提高,出现了很多通过手工方式无法完成测试的情况,尽管在一些大型软件的开发过程中,人们尝试编写了一些小程序来辅助测试,但是它还是不能满足大多数软件项目的统一需要。于是,很多测试实践者开始尝试开发商业的测试工具来支持测试,辅助测试人员完成某一类型或某一领域内的测试工作,测试工具逐渐盛行起来。人们普遍意识到,工具不仅是有用的,而且要对今天的软件系统进行充分的测试,工具是必不可少的。测试工具可以进行部分的测试设计、实现、执行和比较的工作。通过运用测试工具,可以达到提高测试效率的目的。测试工具的发展,大大提高了软件测试的自动化程度,让测试人员从烦琐和重复的测试活动中解脱出来,专心从事有意义的测试设计等活动。采用自动比较技术,还可以自动完成测试用例执行结果的判断,从而避免人工比对存在的疏漏问题。设计良好的自动化测试,在某些情况下可以实现“夜间测试”和“无人测试”。在大多数情况下,软件测试自动化可以减少开支,增加有限时间内可执行的测试,在执行相同数量测试时节约测试时间。而测试工具的选择和推广也越来越受到重视。

在软件测试工具平台方面,商业化的软件测试工具已经很多,如捕获/回放工具、Web 测试工具、性能测试工具、测试管理工具、代码测试工具等,具有代表性的有 IBM 公司的 Rational 测试工具、HP 公司的 LoadRunner 测试工具等。在开放源码社区中也出现了许多软件测试工具并得到了广泛应用,具有代表性的有 Junit、JMeter 等测试工具。

这些测试工具有的用以检查规格说明书的一致性和完整性,有的用以检查编码的静态特征,也有些则支持已提出的测试方法。有关动态测试的工具的开发更为普遍,其中的

一部分工作力图自动产生测试数据，例如 SMOTL 测试编译程序的工具，以及一些支持符号测试的工具 SELECT、DISSECT、ATTEST 和 SADAT 等。由于模块测试时需要用到驱动模块和庄模块，一些模拟测试环境的工具出现了，如R. G. Hamlet设计的 Testmaster 系统和 D. J. Pauzl 设计的 AUT。

近年来，对软件测试理论的研究和测试新方法的探讨一度陷入困惑。人们将注意力更多地转向了软件测试工具，促使测试工具进一步推陈出新。C. Wilson 和 L. J. Osterweil 不久前提出了支持 C 程序数据流分析工具 Omega，主要解决了 C 程序中指针引用的测试问题。E. F. Miller 设计了交互式测试环境 ITB。接着出现的有 M. A. Hennel 和D. Hedley 开发用于 PFORT 语言程序的 LORA 软件测试环境以及 B. Montel 等人设计测试 Petri 网的软件包 OVIDE。D. J. Reifer 对软件测试工具的特点和使用范围作了说明，这对于了解测试工具的全面情况很有帮助。

在使用过程中，人们意识到了：推陈出新的测试工具起到了不可否认的巨大作用，但更重要的是如何利用好这些先进的测试工具，做好测试工作，那就是不应当只在开发结束阶段才集中进行大量的测试，而应该把测试工作贯穿于质量管理的全过程。只有将测试工具与 RUP 方法论结合起来，才是真正的软件测试实践。

1.7　当今测试行业状况

为了同国际 IT 行业接轨，中国软件测试行业近几年有了飞速的发展，以前人们认为的“重开发、轻测试”倾向有了很大的改观。而所需的测试人才却跟不上测试行业的快速发展，测试人员的缺口也越来越大。近些年软件测试人才的缺口超过 30 万，据专家预测，在未来 5 到 10 年中这一数字还将继续增大。中华英才网的招聘数据显示，IT 行业国内外巨头正在加紧争夺软件测试人才，华为一次抛出招聘 50 名软件测试人员的单子，而联想、用友、瑞星等企业也纷纷打出高薪招聘软件测试人才的启事。出人意料的是，收到的简历尚不足招聘岗位数的 50%，而合格的竟不足 30%。

统计显示，在中国 120 多万软件从业人员中，真正能担当软件测试职责的不超过 5 万人。在软件业发达国家，软件测试人员与开发人员之比接近 1∶1，而在中国该比例仅为 1∶8 左右，很多企业没有专门的测试职位。随着中国加入 WTO 后，汽车、电子产品等都有了飞速的发展，也将促使软件测试行业快速发展。

新华网一则新闻

测试结果说了算：N-Gage 历尽波折终推出。

日前，全球第一大手机制造商诺基亚终于调试完 N-Gage 软件平台，正式宣布启动 N-Gage在线游戏服务，意图借此进军手机互联网，在手机游戏市场分一杯羹。早在 2007 年，诺基亚就计划推出 N-Gage 在线游戏服务，但由于软件平台在测试中表现并不稳定而一再跳票(跳票一词主要用于游戏、电影、数码产品领域中，指厂商不能够按照计划按期推出产品及宣布延期)。

“诺基亚一再推迟发布时间，正是为了通过测试检验软件缺陷，减少漏洞，保证服务质

量，保障企业的经济效益。"计算机教育专家谭浩强教授介绍，"通常一个软件刚刚开发出来，或多或少会存在一些缺陷，这是我们无法回避的事实。开发商要满足现代用户的各种应用要求，保证软件能够稳定运行并安全可靠，降低风险，最有效的办法之一就是在软件上线前进行规范的测试。"他还说，软件测试就是对软件产品进行测试和检验。通过必要的测试，软件缺陷数可至少降低75%，而软件的投资回报率能达到350%。

如此巨大的产业效益，使得国外大型软件企业都非常重视软件测试。国外成熟软件企业，软件测试人员和软件开发人员配比大多为1∶1甚至2∶1，例如在Windows 2000的开发团队中，微软配备了250多个项目经理、1700多个开发人员，内部测试人员则达到3200名。而此次诺基亚也动用了1000多名测试工程师对N-Gage手机在线游戏软件平台进行检测。

从市场需求来看，严重的供需失衡的局面促使我国软件测试工程师处在一个地位高、待遇高的"双高"地位。谭浩强教授说："软件测试人才需求的加大，是我国软件行业的产业升级所决定的。由于我国的软件行业目前突破了作坊时代，由前软件开发的单打独斗升级为工业化、流水线式的生产模式。作为工业化的产品，软件测试也就成为软件开发企业必不可少的质量监控部门。而目前我国的软件测试人才的培养较产业升级相对滞后，这就形成了测试人才的供给远小于需求的现状。"

为了吸引更多的人才，企业纷纷采取高薪策略。据统计，测试工程师的起薪在国内从3000～5000元/月不等。工作2～3年后的薪资更是翻番。然而尽管如此，软件测试从业人员不论是数量还是质量，都远远不能满足当今社会的需求。

1.8 测试角色

戏剧舞台上的生、旦、净、丑是不同的角色，其表演方式各自具有明显的特征，这是由于角色决定的。同样，软件测试工程师的角色，在软件项目开发中也存在如何定位和表现自身的行为和责任的问题。须知角色不明，责任不清，行为就失去了参照目标，其结果就可想而知了。轻则降低了工作质量和效率，重则被视为工作能力低下，只能退出软件测试项目组的舞台。

角色决定工作内容和承担的任务。测试工程师的角色应该承担什么任务呢？在传统软件开发周期里，软件测试往往安排在最后，因此对软件测试角色是按照测试工程师的发展过程划分的，分别为初级测试工程师、测试工程师、高级工程师。在从不断学习测试技能，到熟悉软件开发流程，最后到管控整个测试流程的过程中，由测试的执行者向测试的管理者不断提高。提到软件测试工程师，很多人就会想到那些反复使用软件，试图在频繁操作中寻找到错误发生的低层次人员或者软件用户，其实这是一种错误的概念，软件测试早已超越了用户使用来发现Bug的基本测试阶段。

RUP已经清晰地定义了测试人员的不同身份所对应的不同职责。测试人员包括测试分析员、测试经理、测试设计员以及测试人员。沃特金斯在《实用软件测试过程》一书中还划分了其他更多的独立职位，例如测试组长、独立测试观察员、测试分析员、测试自动化

架构师、自动化分析员、探索测试者、测试计划者等。这些职位都是重叠兼职的，并没有必要进行过于详细的划分。

RUP 中有四个角色关注与测试相关的活动：

① 测试经理主要总体负责测试正确和成功地进行，确保对测试资源进行计划和管理，评价测试的过程和效果。

② 测试分析员主要负责识别和定义测试需求，对测试过程和测试结果的详细情况进行监控，评估测试活动结果的质量。

③ 测试设计员是测试中的主要角色。该角色负责生成测试计划和测试模型，执行测试过程，评估测试范围和测试结果及测试的有效性，生成测试评估摘要。

④ 测试员负责执行测试，设置和执行测试，评估测试执行过程并修改错误。

这些角色代表了关于技能和责任的正常而完整的分工。对于一个小型组织中的测试人员，可能会由一个人同时担任这四个角色。而在大型的组织中，这些角色就可能会分别由不同的人员来担当。有了这些人员的分工合作，软件测试就能有条不紊地在可管理的状态下完成。

角色案例

与 RUP 测试角色稍微有点区别，微软的软件测试工程师分为三种：

① 测试执行者(Basic Software Tester)对应 RUP 角色的测试员和测试分析员；

② 测试工具软件开发工程师(Software Development Engineer in Test)对应测试设计员；

③ 高级软件测试工程师(Ad_hoc Tester)对应测试经理。

测试执行者负责理解产品的功能要求，然后根据测试规范和测试案例对其进行测试，检查软件有没有错误，决定软件是否具有稳定性，属于最低级的执行角色。

测试工具软件开发工程师负责写测试工具代码，并利用测试工具对软件进行测试；或者开发测试工具为软件测试工程师服务。产品开发后的性能测试、提交测试等过程都有可能要用到开发的测试工具。对技术要求最强的是测试工具软件开发工程师，因为他们要具备写程序的技术。

高级软件测试工程师属于比较有经验，自己会找方向并做得很好的测试工程师，要求具有很强的创造性。

在开发管理上，测试不归属于项目管理，也不归属开发人员。这三个部门并驾齐驱，相互协作。测试工程师最终决定产品是否能够发布。

担当这些角色需要具备什么条件？或者说什么样的人才能成为测试人员？软件测试工作对软件测试人员水平的要求说高很高，说低也低。水平低的只会修改测试案例，用测试工具进行测试。水平高的可以做项目可行性可靠性分析、风险分析，做测试计划，做测试自动化开发。RUP 对测试人员提出了应具备的素质，测试员和测试经理的素质要求各不一样。

① 测试员的素质要求：

- 掌握有关测试的方法和技术；

- 必备当前测试领域、系统或应用程序等方面的知识；
- 必备有关网络和系统体系结构的知识；
- 接受过使用测试自动化工具的培训；
- 拥有使用测试自动化工具的经验；
- 拥有编程经验；
- 具备调试和诊断技能。

② 测试经理的素质要求：

- 具备软件开发过程各个方面的基本知识；
- 拥有测试方法、技术和工具等方面的经验；
- 有计划和管理才能；
- 有人际交往能力，特别是沟通和领导才能；
- 具备当前测试领域、系统或应用程序等方面的知识（必备）；
- 拥有编程或管理编程团队的经验（必备）。

可以看得出来，测试员的要求比测试经理要求低一些，基本上只要懂得测试技术就可以了。这里就测试经理须具备的素质介绍如下：

具备软件开发过程各个方面的基本知识，这是因为作为一名测试工程师，不能仅从使用者的角度来测试软件产品，而且还要从技术的角度来设计测试用例，这里所说的技术包括基础的与专业的，基础方面应学习过以下的课程并掌握其知识：软件技术基础、C 语言、面向对象设计、C++、数据库理论、计算机网络技术、软件工程、数据结构与算法、离散数学等。专业方面应掌握：软件测试技术概论、测试管理、测试工具、软件质量管理等。这些知识可能会在软件开发过程中用到。

计划和管理才能是指执行任何任务都要制定计划，把各项任务按照轻、重、缓、急列出计划表，分配下属来承担，自己把眼光放在部门未来的发展上，不断理清明天、后天、下周、下月，甚至明年的计划上。在计划的实施及检查时，要预先掌握关键性问题，不能因琐碎的工作，而影响了应该做的重要工作。要清楚做好 20% 的重要工作，等于创造 80% 的业绩。

管理才能是指考虑问题要全面，结合客户需求、业务流程和系统构架等多方面考虑问题。无论计划如何周到，如果不能有效地加以执行，仍然无法产生预期的效果，为了使下属有共同的方向可以执行制定的计划，适当地管理是有必要的。管理下属，首先要考虑工作分配，要检测下属与工作的对应关系，也要考虑管理方式，语气不好或是目标不明确，都是不好的管理。而好的管理可以激发下属的热情，而且能够提升其责任感与使命感。要清楚管理的最高艺术，是下属能够自我指挥。

具备人际交往能力，尤其应具备沟通技巧和领导才能。这是因为测试经理需要与很多人员进行沟通，他的很多时间都要花在沟通工作上。沟通不仅包括内部上下级、部门与部门之间的协调，也包括与外部客户、关系单位、竞争对手之间的利益协调，任何一方沟通不好都会影响执行计划的完成。面对不同人员，需要不同的语气、不同的态度。对客户就要有亲和力，处处为客户着想，客户就是上帝。对协同工作的人员更要讲究沟通技巧。有人说，开发项目经理、开发人员、客户、市场人员等都是测试经理经常面对的对象。这说明

了项目经理需要具备很强的沟通协调能力。如果缺乏沟通技巧，就会形成经常吵架的局面。测试经理在说话的语气或讲述一个问题的出发点时要特别注意。要与开发小组很好地沟通，试着给他们找一个“Bug 杀手”，或对他们说“我简直不敢相信，你写的程序居然到现在没有找到 Bug”。

领导才能包括控制和授权能力。

- 控制就是追踪考核，确保目标达到、计划落实。虽然谈到控制会令人产生不舒服的感觉，然而企业的经营有其十分现实的一面，有些事情不及时加以控制，就会给企业造成直接与间接的损失。但是，控制若是操之过急或是控制力度不足，同样会产生反作用：控制过严使下属口服心不服，控制不力则可能现场的工作纪律也难以维持。要清楚最理想的控制，就是让下属通过目标管理方式实现自我控制。
- 任何人的能力都是有限的，作为高级经理人不能像业务员那样事事亲历亲为，而要明确自己的职责就是培养下属共同成长，给自己机会，更要为下属的成长创造机会。下属是自己的一面镜子，也是延伸自己智力和能力的载体，要赋予下属责、权、利，下属才会有做事的责任感和成就感，一个部门的人琢磨事，肯定胜过自己一个人琢磨事，这样下属得到了激励，自己又可以放开手脚做重要的事，何乐而不为。切记成就下属，就是成就自己。

具备当前测试领域、系统或应用程序等方面的知识，这是因为软件以服务于用户为目的，必然要适合用户的逻辑思维和认知过程。一名合格的测试人员不仅要熟悉软件测试的技术和理论，更要熟悉用户专业领域的知识，并能够将二者有机地结合。例如测试财务软件，测试员起码应懂得会计知识。另一方面，不同产品的特性不一样，对测试工具要求也是不同的，例如 Windows 的测试工具不能用于 Office，Office 的也不能用于 SQL Server，微软很多测试工程师就是负责专门为某个产品写测试程序的。

拥有编程或管理编程团队的经验，这是因为测试工程师有时候需要对源码进行检查，有时候也会从程序结构的角度来测试软件，有时候需要写一些自动测试的工具软件，有时候需要写测试脚本，显而易见，会写简单代码，能读懂源码对测试人员来说是必须的，而且如果有一定的编程经验，可对软件开发过程有较深入地理解，并从编程人员的角度来正确地评价。

其实 RUP 对测试角色只列出了看得见摸得着的素质要求，还有反映在素质后面的要求没有列出来。那就是创新意识和质疑态度。

创新是衡量一个人、一个企业是否有核心竞争能力的重要标志。要时时、处处、事事都有强烈的创新意识。这就需要不断地学习，善于思考。而这种学习与大学里那种单纯以掌握知识为主的学习是不一样的，它要求把工作的过程本身当作一个系统的学习过程，不断地从工作中发现问题、研究问题、解决问题。解决问题的过程，也就是向创新迈进的过程。因此，做任何一件事都要认真想一想，有没有创新的方法使执行的力度更大、速度更快、效果更好。要清楚创新无极限，唯有创新才能生存。

创新能力案例

微软总部测试经理陈宏刚博士曾讲过这样一个故事：

刚进入微软时，老板也是只给陈宏刚一个操作流程，每天就按照这个规程去做，几天

下来,一个 Bug 都没有发现。陈宏刚也很沮丧,觉得这样挺对不起公司,后来自己问自己:为什么非要这样做!于是换了其他的方法试试,令他吃惊的是,一下就找到很多严重的 Bug,当时也不敢声张。有一天,他找到 10 多个非常严重的 Bug,开发经理一下就惊呆了,怒气冲冲地跑到陈宏刚面前问:“你是不是改变了测试方式和测试步骤?”陈宏刚被吓住了,说:“可能改变了一点”。对方说:“我非常生气,但我不是生你的气,而是因为以前测试人员水平太差,或者以前的测试方法有问题,软件中有些 Bug 存在了半年甚至一年,直到现在才被发现,我有亡羊补牢的感觉,而且现在修补这些错误比原来就困难多了”。后来陈宏刚得到了老板的赞许,可以按照自己的想法去做测试。对此,陈宏刚感受颇深:“一方面微软非常鼓励创造的文化,同时也感到循规蹈矩只守教条的测试人员就不是好的测试人员,只是和一般用户一样了。做软件测试工程师同样需要开拓和创造性”。

软件的使用者千差万别,软件在使用过程中遇到的各种问题也是千差万别的,所以要求软件测试工程师需要具有逆向思维和质疑态度。这是作为一名优秀的软件测试工程师最基本的素质。因为任何事情不可能绝对正确,错误、缺陷总是存在,只有具备逆向思维能力,才能够发现和找到缺陷的隐身之处。除了漏洞检测,测试还应该考虑性能问题,也就是要保证软件运行得很好,没有内存泄漏,不会出现运行越来越慢的情况;在不同的使用环境下,考虑软件的兼容性也同样重要;即使是测试同样的软件产品,而规模大小也决定其寻找错误的目标,因为软件的 Bug 往往在大型软件的连接处。做测试要考虑到所有出错的可能性,还要做一些超出常规常理的事。测试人员不能总是以常规的思路来测试软件,有时要设计一些非常规的、相反的测试用例来不断地测试软件产品,进行破坏性地测试。

逆向思维案例

微软总部有个 40 多岁的测试经理就是原来的家庭主妇,说起来也许大家不会相信。

这名家庭主妇是一位海军军官的妻子,三个孩子的母亲。她只读到高中毕业,连大专都没有上(在美国没有上过大专的人是很少的)。她使用计算机的水平也非常初级,而且还是跟着自己的女儿学的。后来她在家闲得无聊,就决定出来找一个工作,后来居然跑到微软应聘。

当时微软招聘经理在面试她的时候就已经发现她的计算机水平很有限,只能达到一个一般用户的标准。但是招聘经理还发现她的思维很怪异,怪点子很多,能够很快地发现一些问题。于是就让她试用一下 IE,结果她当场就找出了好几个 Bug。后来微软招聘经理对老板说,我想雇用她。老板一听,睁大了双眼:“你疯了吗?你居然想雇用一个家庭主妇!”于是招聘经理跟他说了一下想法。老板还是觉得让一个大学都没有上过的家庭妇女做测试人员是不可思议的。但是,最后他还是说:“你是招聘经理,还是你自己来决定吧!”

微软招聘经理最终决定雇用她。在开始阶段,她的确存在许多问题。招聘时只是了解到她的逆向思维很强,却忽视了她的其他素质。由于她一直做家庭妇女,没有在职业环境中待过,因此显得很粗鲁,经常大声喧哗,利用办公电话大声打电话,在办公室里抽烟,而且还经常逞能,走到别人后面得意地告诉别人:“我刚刚找到一个 Bug!”,好像别人都

发现不了Bug。但是警告过她以后，她就努力改正了这些毛病，并非常认真敬业。她学得非常快，三个月以后，就已经非常专业了。最后，微软老板终于承认她非常厉害，并将她转为了正式职员。同时微软也招聘了一些学生物、物理专业的博士，但他们做测试工作做得并不怎么样，远不如这位家庭妇女干得好。

这个例子说明测试工作不一定要非常懂开发（或者开发出身），但是懂一些开发会对个人深入理解测试有帮助，专业学习除了参加培训外，还需要在工作中不断地学习，不断地总结。

做软件测试工程师需要对软件抱有怀疑态度。这是因为软件测试要准备各种数据，从每个细节上设计不同的应用场景，不能想当然地假定任何一个数据是可行的。并且不要停止怀疑，要有"打破砂锅问到底"的精神，对于只出现过一次的Bug，一定找出原因，不解决问题誓不罢休。

例如：有人问阿凡提："我肚子痛，应该用什么药？"阿凡提说："应该用眼药水，因为你眼睛不好，吃了脏东西才肚子痛。"按照常规常理，应该是"脚痛治脚，肚子痛治肚子"，而阿凡提却能够顺藤摸瓜，运用归纳、推理等方法，去找生病的根源。测试人员所要具备的就是这种思维能力，来寻找、判断错误的根源。

1.9 职业规划

软件测试工程师发展有几种途径：

第一种走技术路线，成长为质量保证经理（SQA），这时他能够独立测试很多软件，再向上可以成为软件测试架构设计师、技术专家。

第二种就是向管理方向发展，从测试工程师到组长，再到项目经理，到更高的职位，可以作职业经理人，也可以自己创业。

第三种可以换职业，做项目管理，做开发人员都可以，很多测试工具软件开发工程师在写测试软件的过程中，因为开发方面积累了经验，同时对软件产品本身产生了自己的看法，很容易转去做产品编程。

一般国内常见的软件测试人员成长之路如下：

① 测试员→测试工程师→测试主管→质量保证经理；

② 测试员→程序员→开发主管→质量保证经理。

软件测试员的一生如同一名医生的一生，随着职业阅历和临床经验的丰富累积，到一定的年龄，他们可以通过"望闻问切"就能知道毛病出在什么地方。因此，有人说软件测试员和医生一样，不需要用"青春"来保证和延续自己职业寿命。

一般来说，传统IT技术人员，由于专业特点和长期的职业生涯所形成的个性等因素的制约，很难转岗，上升空间也相对受限。而软件测试则不同，由于工作的特殊性，测试人员不但需要对软件的质量进行检测，而且他们的工作涉及对软件项目的立项、管理、售前、售后等领域。在这个过程中，他们不仅提升了专业技能，还有项目管理、沟通协调、市场需求分析等能力都得到了很好的锻炼，从而为自己的多元化发展奠定了基础。而且，由于这

个行业人才稀缺，具备专业技能的软件测试人员经过几年实践后，很容易得到晋升。因此，很多 IT 行业的从业人员，也选择了通过做软件测试拓展新的发展空间。

随着软件测试行业的职位不断细化，每个人在自己擅长的领域走向深入，都可以成为该领域的技术专家。所谓技术专家，要在自己经营的领域里，具有个人独到的见解和深厚的技术实力，而这类人才可以不再从事具体的测试工作，而是提供行业性测试技术咨询、培训等，为软件测试整个行业的发展，起到了不可低估的带头作用。在一些专业的咨询、培训公司，或者 IBM、Microsoft 等巨型公司，不乏这样的人才；然而目前在我国，这样的人才不多，这为我们提供了努力的方向。

一方面，由于软件测试人才更强调经验积累，在几年的测试经验背景下，可以逐步转向管理或资深测试工程师，担当测试经理或 QA 部门主管，所以有发展空间，且职业寿命更长久；另一方面，由于国内软件测试工程人才奇缺，各种类型软件企业都需要这方面人才，所以学习测试专业的好找工作，且待遇普遍较好。

质量是产品的灵魂，软件测试是软件产品质量的保证。软件测试工作的重要作用在软件产业中任何时候都是不可替代的，软件测试工作包含了技术及管理的各个方面，对年龄的要求也没有一定的限制。因此，在竞争越来越激烈的 IT 职场中，软件测试工程师的工作相对来说更稳定、更有发展前景。

习题与思考

1. 为什么说软件开发发生了本质的变化？
2. 造成软件错误的原因有哪些？
3. 软件测试第一类方法和第二类方法有哪些？
4. IEEE 给软件测试下的定义是什么？
5. 软件测试模型经历了哪些演变？
6. 常用的软件测试有哪些类型？
7. 软件测试工具有哪些？
8. 什么叫真正的软件测试实践？
9. RUP 有哪些角色？需要具备什么素质？
10. 为什么说软件测试工程师的工作更稳定、更有发展前景？



第2章 RUP基础

福特的故事

1908年以前，轿车在美国都是昂贵的消费品，因为当时的轿车需要技术熟练的工人根据技术图纸将上万个轿车配件例如轮胎、螺丝等组装起来。针对此弊端，福特设计了T型流水线，该流水线有84个不同的步骤，每个工人通过培训只完成一个步骤。这个创新将原先装配底盘所需的12个小时30分钟的时间减少到2个小时40分钟，由原先850美金一辆车的价格降至360美金。福特也由此成为全美盈利率最好的轿车产商。

软件开发过程由方法论和工具构成(过程＝方法＋工具)。从福特的故事可以看到，只要有工具和技术就可以胜任轿车装配任务。但为了减少失误、保证质量和提高效率，人们往往采用流水线作业，正如生产福特轿车一样，流水线作业便是一种应用于福特轿车装配中的方法论。

开发一个具有一定规模和复杂性的软件系统和编写一个简单的程序大不一样。其间的差别，借用RUP创始人之一G. Booch的比喻，如同建造一座大厦和搭一个狗窝的差别。大型的、复杂的软件系统的开发是一项浩大的工程，必须借鉴福特轿车装配中的方法论，按工程学的方法组织软件的生产与管理，必须经过分析、设计、实现、测试、维护等一系列的软件生命周期阶段。这是人们从软件危机中获得的最重要的启示。

软件开发就像建造一座宏伟的宫殿，从计划、设计到施工，每一个环节都必须严格把关，稍有不慎，整个工程就会失败。据统计，美国每年就有180 000个信息技术项目，耗资大约2500亿美元，其中大约有25％～30％的项目会流产。由此可见，由于管理不善和设计上的失误所造成的损失是巨大的。现代软件开发的管理和方法论显得比以往任何时候都更为重要。

目前，信息技术市场流行的方法论有RUP、XP(Extreme Programming)等。在这些方法论中，最流行的要数RUP。RUP是IBM公司的一套软件开发过程产品，是目前影响较大的、面向对象的软件开发过程。RUP提出了一整套以UML为基础的开发准则，用以指导软件开发人员以UML为基础进行软件开发。UML的全称是Unified Modeling Language，即统一建模语言。UML的目标是为开发团队提供标准通用的设计语言来开发和构建计算机应用，它提出了一套IT专业人员期待多年的统一的标准建模符号。通过使用UML，这些人员能够阅读和交流系统架构和设计规划——就像建筑工人所使用的建筑设计图一样。因RUP与当前流行的Java、J2EE技术和面向对象的设计

思想（OOAD）紧密地结合在一起，所以在大型的信息技术项目中得到了广泛的应用。

学习本章的目的是使读者能对 RUP 理论建立起概要性、框架性的整体认识，并为后续章节测试技术的学习打好基础。具体讲，就是读者要在头脑里建立起 RUP 的框架，即一个生命周期、三个理解、三个特点、六个经验、六个原则。其中重点是三个理解和三个特点。

2.1　RUP 的发展史

20 世纪 80 年代末至 90 年代初，面向对象方法学进入鼎盛时期，当时比较著名的面向对象方法学已有 50 多种。其表示法和分析设计方法各有特点，但是，这些方法学具有一定的一致性和兼容性。于是，James Rumbaugh、Grady Booch 以及 Ivar Jacobson 决心吸收各家所长，创建一个统一的方法学。最终在他们努力和推动下，诸多面向对象的方法学在表示法和分析设计理论上达成了一致，并先后推出了面向对象开发的行业标准语言 UML 和 RUP 理论。这三位面向对象领域的杰出专家还创建了 Rational 公司，致力于推广 RUP 理论，并领导 Rational 公司占领了全球 70%以上的市场份额。2002 年 12 月 6 日，IBM 公司宣布将以 21 亿美元现金收购 Rational 软件公司，此后 Rational 公司并入 IBM 公司麾下。由于 Rational 在面向对象方法学领域的强大号召力，Rational 品牌保留至今。图 2-1 这幅漫画形象地描绘出了这个软件方法学上最为重要的统一。

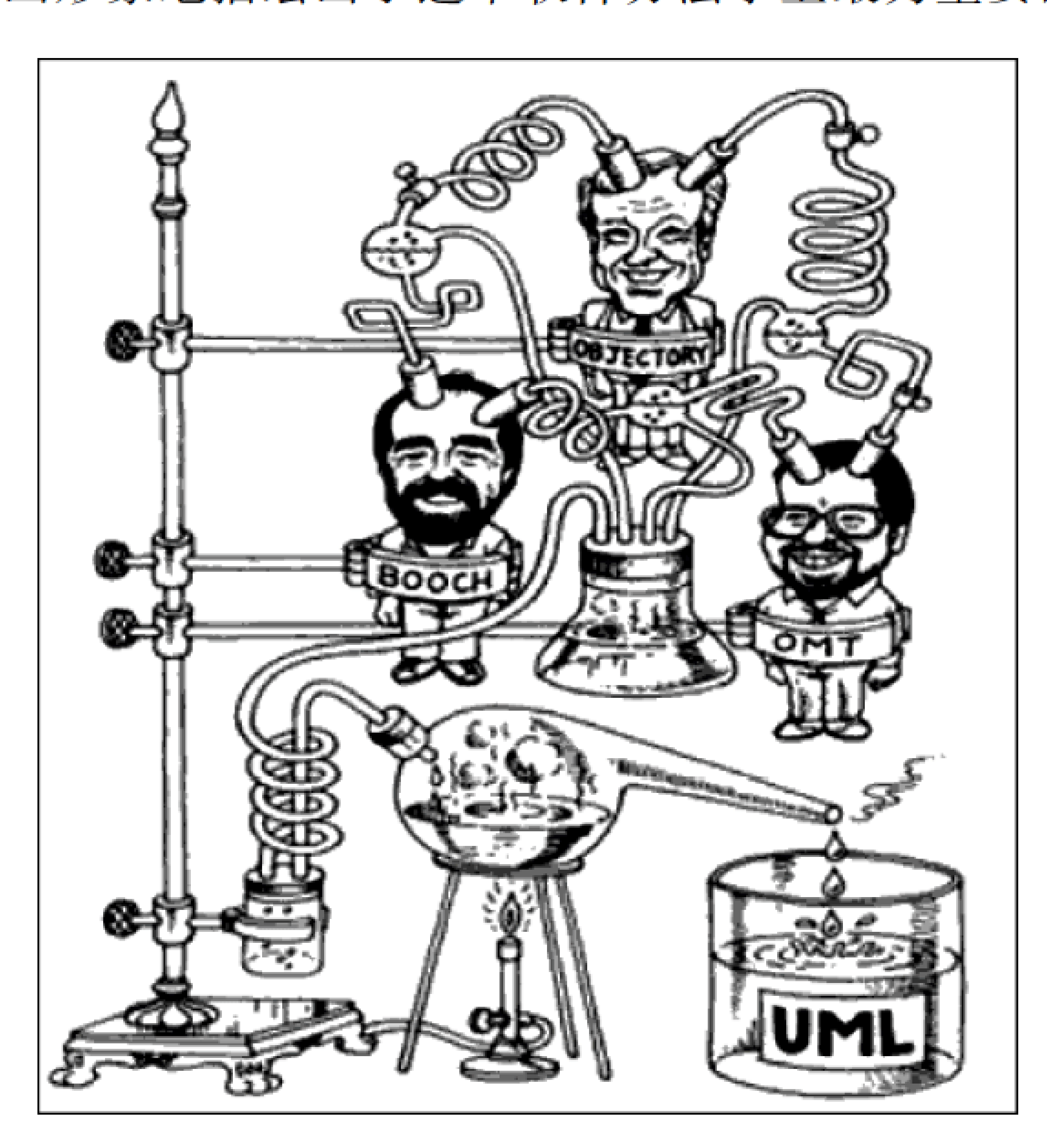


图 2-1　关于 UML 的漫画

作为 Rational 公司推出并维护的一个软件过程产品，RUP 从“Ericsson（爱立信）方法”（1967 年）开始，到“对象工厂过程”（1987—1995 年），再到“Rational 对象工厂过程”

(1996—1997 年)，直至最后的"Rational 统一过程"(1998 年至今)，经过了三十多年的发展历程，现最新版本为 RUP 2007。目前，全球有上千家公司在使用 Rational 统一过程，如 Ericsson、MCI、British Aero Space、Xerox、Volvo、Intel、Visa、Oracle 等，它们分布在电信、交通、航空、国防、制造、金融、系统集成等不同的行业和应用领域，开发着或大或小的项目，这表现了 Rational 统一过程的多功能性和广泛的适用性。

2.2 什么是 RUP

RUP 的全称是 Rational Unified Process，即统一软件过程，它是从已形成的各种面向对象分析与设计方法中吸取精华，为软件开发人员以 UML 为基础进行软件开发提供了一个普遍的软件过程框架，可以适应不同的软件系统、应用领域、组织类型和项目规模。它提供了在开发组织中分配任务和职责的严格方法。它的目标是按照预先制定的时间计划和经费预算，开发高质量的软件产品，以满足最终用户的需求。

可以从以下三个方面来理解 RUP。

1. RUP 是一套面向对象方法学

RUP 是以面向对象方法为基础的方法学，在业务建模、需求、分析设计、实现、测试等各个规程中，面向对象方法贯穿始终。即便是在与系统构建关系不大的业务工程中，RUP 也使用面向对象方法。

RUP 坚持以用例(use case)驱动，以架构为中心，迭代和增量的开发方法(见 2.3 节)。用例驱动既做到了以客户为中心，从客户的角度看系统，为客户创建真正可用的系统的构造方式。以架构为中心，坚持综合考虑软件系统的各个方面，并优先解决各个方面的主要问题，同时创建常见问题的通用解决方案，从而降低了软件项目的主要风险，准确估算项目进度，提高软件复用、迭代和增量式开发，以风险为驱动，分阶段针对不同的风险制定对策，以保证软件项目的成功完成。

2. RUP 是一种适用范围较广的适应性软件过程

RUP 定义了进行软件开发的工作步骤，即定义了软件开发过程中什么时候做，做什么，怎么做，谁来做的问题，以保证软件项目有序地、可控地、高质量地完成。

RUP 是一种适应性软件过程，有别于瀑布模型类的预见性软件过程。RUP 不假设从一开始就可以掌握软件开发的全过程，而是坚持以迭代方式推进软件开发，结合不断演进的项目状态和现实变化做出相应的调整，制定出新的计划。实践证明适应性过程比预见性过程更能保证项目成功。

另一方面，RUP 并没有对软件开发的规范化程度做出明确规定。换言之，RUP 允许开发人员根据项目的实际情况，对其进行裁剪，以决定哪些文档、过程是必需的。

3. RUP 是一个提供了可定制框架的软件过程产品

RUP 提供了一个可定制软件产品,其中包括 RUP 方法学指导、过程定义和文档模板,也还包括一些示例工程。除此之外,为了支持 RUP 的方法学理论和最佳实践,很多软件公司例如 IBM 公司还提供了一系列 CASE 产品,如 RequisitePro(需求管理)、ROSE(可视化建模工具)、ClearCase(配置管理工具)和 ClearQuest(变更管理工具)等。

2.3 RUP 的特点

RUP 有三个显著特点:

(1) 它是一种迭代和增量的软件开发过程;

(2) 软件开发是由用例驱动;

(3) 软件开发是以构架设计为中心的。

2.3.1 迭代和增量开发

迭代(iteration)是按照专门的计划和评估标准产生一个内部的或外部的发布版本,所进行的一组明确的活动。简单地讲,迭代是将整个项目分为许多更小的迷你项目,它更容易管理和成功完成。每个迷你项目是一个迭代,当然每个迭代包含正常软件开发项目的所有元素,即执行需求获取、分析、设计、实现和测试这五个活动。一般一次迭代由四个规程组成:需求、分析设计、实施和测试。

迭代后要对迭代结果进行评价,然后计划并进行下一个小项目,直至完成整个项目。例如开发一个类似 Word 的软件,现在已开发了文件管理模块,正在开发编辑模块,但后来发现,文件管理模块有某些功能还没有实现,可以在编辑模块的开发过程中同时继续开发文件管理模块,如此不断地反复,所以说这个过程是迭代的过程。经过这样的反复迭代后该软件的功能就会越来越完善,最终开发出优秀的产品。

早期迭代要积累需求、迁移主要风险、明确问题、寻找解决问题的知识、进行项目定位、建立体系结构基线和为后续开发制订详细计划,用较少的人力和物力进行此项工作;后期迭代要降低风险、实现组件和产生增量。

名词解释:基线

《IEEE Standard Glossary of Software Engineering Terminology》中对基线的定义为:基线是已经通过了正式复审的规格说明或中间产品,它可以作为进一步开发的基础,并且只有通过正式的变化控制过程才能改变。

以软件需求规格说明为例,开发人员可以在需求开发阶段,随时根据用户的要求修改该文档,一旦该文档通过正式评审形成基线后,需求变更就必须受到严格的限制,原则上是不允许轻易变更的,必须按照规定的变更控制程序进行申请、评估、修改和验证。

增量(increment)指在后续迭代结束后,两个发布版本之间存在的差异或差值。例如在软件开发过程中,先开发主要功能模块,再开发次要功能模块,逐步完善,最终开发出符合需求的软件产品。以上面的开发 Word 为例,应该首先开发出文件管理(保存、读取文件)、基本编辑功能、打印等,而其他不太常用的功能可以最后开发。每一次增量都会添加更多的软件功能——这与往墙上添砖加瓦有几分相似。因此增量是系统中一个较小的、可管理的部分(一个或几个构造块),即一次迭代的结果是一个增量。通过不断地迭代,不但获得增量,尽快获得反馈,而且也在不断地降低风险。

增量迭代是 RUP 的核心,面对当今的复杂的软件系统,如果首先定义整个问题,设计完整的解决方案,编制软件并最终测试产品是不可能的。它需要一种能够通过一系列细化若干个渐进的反复过程,而生成有效解决方案的迭代方法。RUP 支持专注于处理生命周期中每个阶段中最高风险的迭代开发方法,这样可极大地减少项目风险。迭代方法通过可验证的方法来帮助减少风险,经常性的可执行版本使最终用户不断地介入和反馈,因为每次迭代过程以可执行版本告终,开发队伍停留在产生结果上,频繁的状态检查确保项目能按时进行,迭代化方法同样使得需求变化和完善更为容易。

图 2-2 显示了一种描述所使用的迭代方法的方式。它显示了该项目的三个演进,用圆柱来表示迭代。注意到每个圆柱都包含于表示演进的螺旋中。

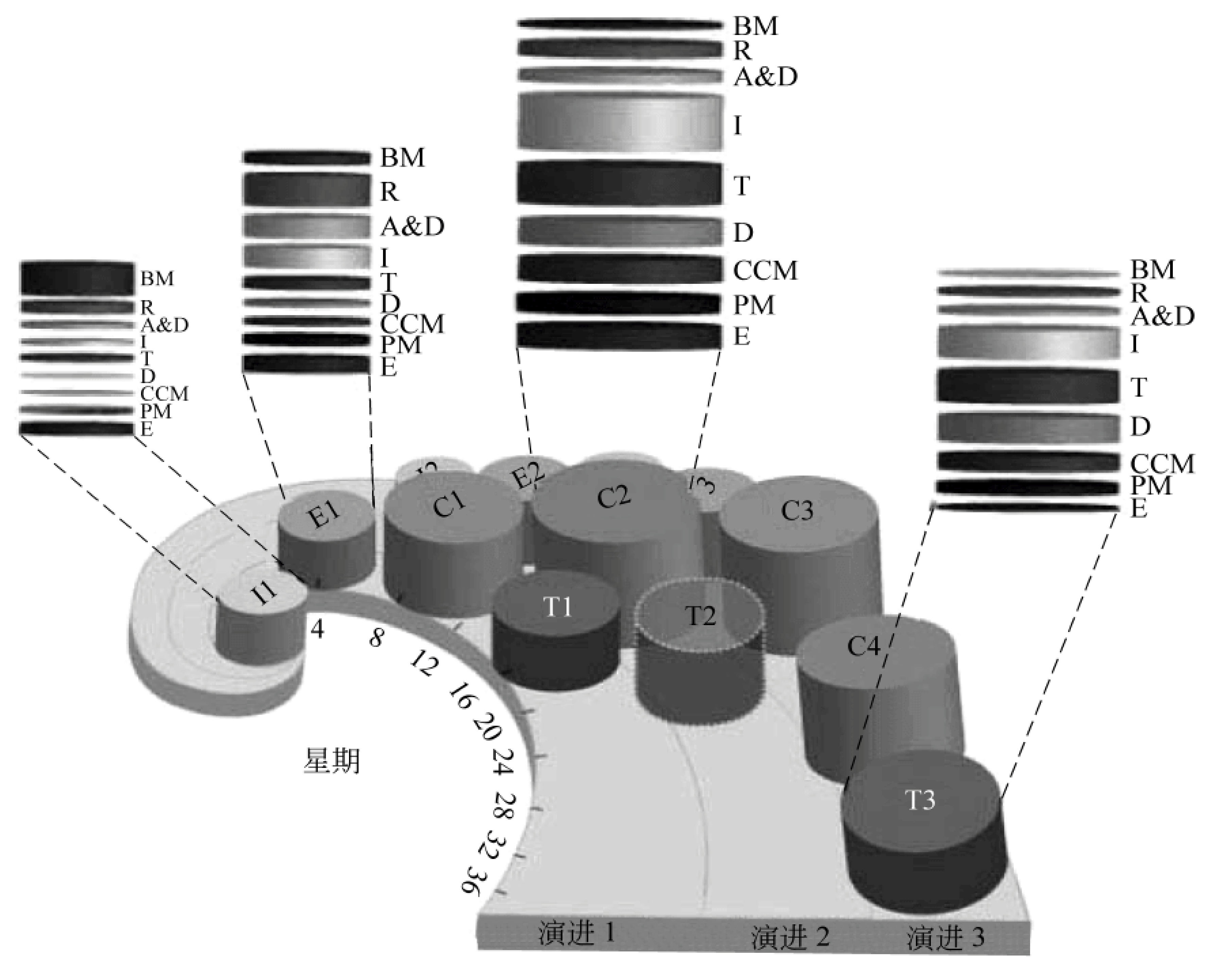


图 2-2 迭代过程

相对于传统的瀑布过程,迭代过程具有以下优点:

- 在早期就降低风险;
- 对变更易于进行控制;

• 更高的可复用性；
• 项目团队在开发过程中可以学习；
• 更好的整体质量。

迭代加快了与用户反馈的步伐，保证最终产品不出现大的偏差。迭代减小了需求、设计、编码各阶段的风险，及时重构和发现问题。

名词解释：重构

重构一般是指通过修改代码或数据，使软件符合新的要求。目的是在不改变“软件的外部行为”的前提下，提高其可理解性，降低其修改成本。因此重构通常并不推翻原有软件的体系结构，主要是改造一些模块和数据结构，让代码更加容易理解，然后更容易维护。重构的一些好处如下：

① 使软件的质量更高，或使软件顺应新的潮流（标准）。

② 使软件的后续（升级）版本的生产率更高。

③ 降低后期的维护代价。

要注意的是，在代码重构和数据重构之后，一定要重构相应的文档。

2.3.2 用例驱动

RUP 的另一大特征是用例驱动。用例是 RUP 中一个非常重要的概念，它描述了用户怎样和系统交互。简单地说，一个用例就是系统的一个功能。例如，在一个基于电子商务的医疗系统中，病人可以坐在家里通过网上浏览器与医生约定看病的时间（Make appointment），这样“Make appointment”就是系统的一个用例。一般在系统分析和系统设计中，将一个复杂的庞大系统分割、定义成一个个小的单元，这个小的单元就是用例。

在系统的生命周期中，以用例为驱动意味着：为建立系统，与系统有关的人员需要进行交流，而将用例作为主要制品。以用例为驱动还意味着：用例是对系统进行分析、设计、实现和测试的基本输入，包括对系统体系结构的创建、验证和确认。即以用例为单位制定计划、分配任务、监控执行和进行测试等，将实现软件开发的核心工作有机地组合为一体，在产品开发的各个阶段中都可以回溯到用户的实际需求。按照 RUP，用例贯穿整个软件开发的生命周期。在商务需求分析中，客户或用户对用例进行描述，在系统分布和系统设计过程中，设计师对用例进行分析，在开发实现过程中，开发编程人员对用例进行实现，在测试过程中，测试人员对用例进行检验。因此 RUP 是一种以用例为中心的开发过程，而基于 RUP 的软件测试也是以用例为基本依据进行的。

通过用例，可以得到体系结构和其他制品。首先选择在体系结构上具有重要意义的用例，接着实现那些关键的用例，构造出系统的初步体系结构，逐步得到稳定的体系结构。通过枚举用例的不同执行路径，可导出测试案例和测试规程。通过用例，还可估算系统性能、硬件需求和可用性，并能进行用户界面设计，也可以把用例作为编写用户手册的基础。

对用例的细化要在一定的程度上涉及系统的内部功能，对各用例进行完整的功能描述。细化时要区分用例中的三类事务：反映系统与参与者（actor）之间的接口，在用例中哪些包含实现接口（功能）的活动和属性，以及对前两者进行协调和控制的机制。以用例

为驱动也有助于模型之间的追踪和系统演化。

2.3.3 以构架设计为中心

RUP的第三大特征是它强调软件开发是以构架为中心的。架构是系统在其所处环境中最高层次的概念。软件系统的架构是指通过接口交互的重要组件的组织和结构，这些组件又由一些更小的组件和接口组成。架构与系统可类比为：架构就像躯体的骨架，而系统则是包括了骨架、皮肤和肌肉的整个躯体。华中科技大学教授杨叔子院士曾指出"文学中有科学，音乐中有数学，漫画中有现代数学的拓扑学。漫画家可以几笔就把一个人画活，不管怎么美化或丑化，就是活像。为什么？因为那'几笔'不是别的，而是拓扑学中的特征不变量，这是事物最本质的东西。"他指出了构架是软件系统中最本质的东西。

① 构架是对复杂事物的一种抽象。良好的构架是普遍适用的，它可以高效地处理多种多样的个体需求。一提起"房子"，我们的大脑中马上就会出现房子的印象（而不是山洞的印象）。"房子"是人们对住宿或办公环境的一种抽象。不论是办公楼还是民房，同一类建筑物（甚至不同类的建筑物）之间都具有非常相似的体系结构和构造方式。

② 构架在一定的时间内保持稳定。只有在稳定的环境下，人们才能干点事情，社会才能发展。科学告诉我们，宇宙间万物无时无刻不在运动、飞行。由于我们的生活环境在地球上保持相对稳定，所以我们可以无忧无虑地吃饭和睡觉。软件开发最怕的就是需求变化，但"需求会发生变化"是个无法逃避的现实。人们希望在需求发生变化时，最好只对软件做些皮毛的修改，可千万别改动软件的构架。就如人们对住宿的需求也会变动，你可以经常改变房间的装潢和摆设，但不会在每次变动时都要去拆墙、拆柱、挖地基。如果当需求发生变化时，程序员不得不去修改软件的构架，那么这个软件的系统设计是失败的。

良好的构架意味着普适、高效和稳定。一个应用系统的健壮、稳定、可扩展、可维护以及诸多的功能性需求都需要通过灵活稳定的架构来固化，好的软件架构保证系统的高度重用和弹性结构。RUP支持基于组件的软件开发，组件是实现清晰功能的模块子系统。RUP提供了使用新的及现有组件定义体系结构的系统化方法，它们被组装为良好定义的结构或是特殊的底层结构。

构架设计（Architectural Design）作为系统设计的一个重要组成部分，设计师（Architect）必须完成对技术和运行平台的选取，整个项目的基础框架（Framework）的设计，完成对公共组件的设计，如审计（Auditing）系统、日志（Log）系统、错误处理（Exception Handling）系统、安全（Security）系统等。设计师必须对系统的可扩展性（Extensibility）、安全性（Security）、可维护性（Maintainability）、可延拓性（Scalability）、可重用性（Reusability）和运行速度（Performance）提出可行的解决方案。

构架设计的主要步骤如下：

① 在了解用例之后，得到系统体系结构的粗略纲要（独立于特定用例和平台）。

② 关注关键用例。关键用例是有助于降低最大风险的用例，对系统用户来说是最重要的用例，以及有助于实现所有重要的功能而不遗留任何重大问题的用例。

③ 随着对用例的规约，并考虑软件需求、中间件、遗产系统和非功能性需求等，不断产生更加成熟的用例和更多的系统体系结构成分。

④ 经过多次迭代，得到可执行的体系结构基线。

⑤ 经过逐步演化，形成稳定的系统体系结构描述。

阅读材料

Windows NT 的一位系统设计师拥有 8 辆法拉利跑车，让 Microsoft 公司的一些程序员十分眼红。但只能羡慕而不能愤恨，因为并不是每个程序员都有本事成为复杂软件系统的设计师。系统设计要比纯粹的编程困难得多。即便你清楚客户的需求，却未必知道应该设计什么样的软件系统——既能挣最多的钱，又能让客户满意。“天下西湖三十六，最美是杭州”，一千多年前大文豪苏东坡对西湖精彩绝伦的系统设计的描述，使杭州荣升为“天堂”。

系统设计包括四方面内容：构架设计、模块设计、数据结构与算法设计、用户界面设计。如果将软件系统比喻为人体，那么：

① 构架就如同人的骨架。如果某个家伙的骨架是猴子，那么无论怎样喂养和美容，这家伙始终都是猴子，不会成为人。

② 模块就如同人的器官，具有特定的功能。人体中最出色的模块设计之一是手，手只有几种动作，却能做无限多的事情。人体中最糟糕的模块设计是嘴巴，嘴巴将最有价值但毫无相干的几种功能如吃饭、说话、亲吻混为一体，使之无法并行处理。

③ 数据结构与算法就如同人的血脉和神经，它让器官具有生命并能发挥功能。数据结构与算法分布在体系结构和模块中，它将协调系统的各个功能。

④ 用户界面就如同人的外表一样，具有亲和力。赏心悦目的外表让人一见钟情，冷面妖艳、丑陋作怪的外表让人一见恶心。像人们追求外秀内惠那样，软件系统也追求（内在的）功能强大和（外表的）界面友好。但随着生活节奏的加快，人们很少有兴趣去品味深藏不露的内在美。

在进行系统设计时，我们要注重软件的质量因素，如正确性与精确性，性能与效率，易用性与可理解性，可复用性与可扩充性等。即使把系统设计做好了，也并不意味着就能产生好的软件系统。在程序设计、测试、维护等环节还要做大量的工作，无论哪个环节出了差错，都会把系统搞砸。

2.4 RUP 软件开发生命周期

在 RUP 中，系统的生命周期由一系列的周期组成，每个周期产生的结果是用户产品的发布。第一次经历四个阶段被称作初始生命周期，后面的每一次对系统的演化而经历的四个阶段都被称作演化周期。这个周期是一个二维的软件开发模型，如图 2-3 所示。

在图 2-3 中，水平轴代表时间，显示了过程动态的一面，是用周期（cycle）、阶段（phase）、迭代（iteration）、里程碑（milestone）等术语来描述的。垂直轴代表过程静态的一面，是用活动（activity）、产品（artifact）、工人（worker）和规程描述的。

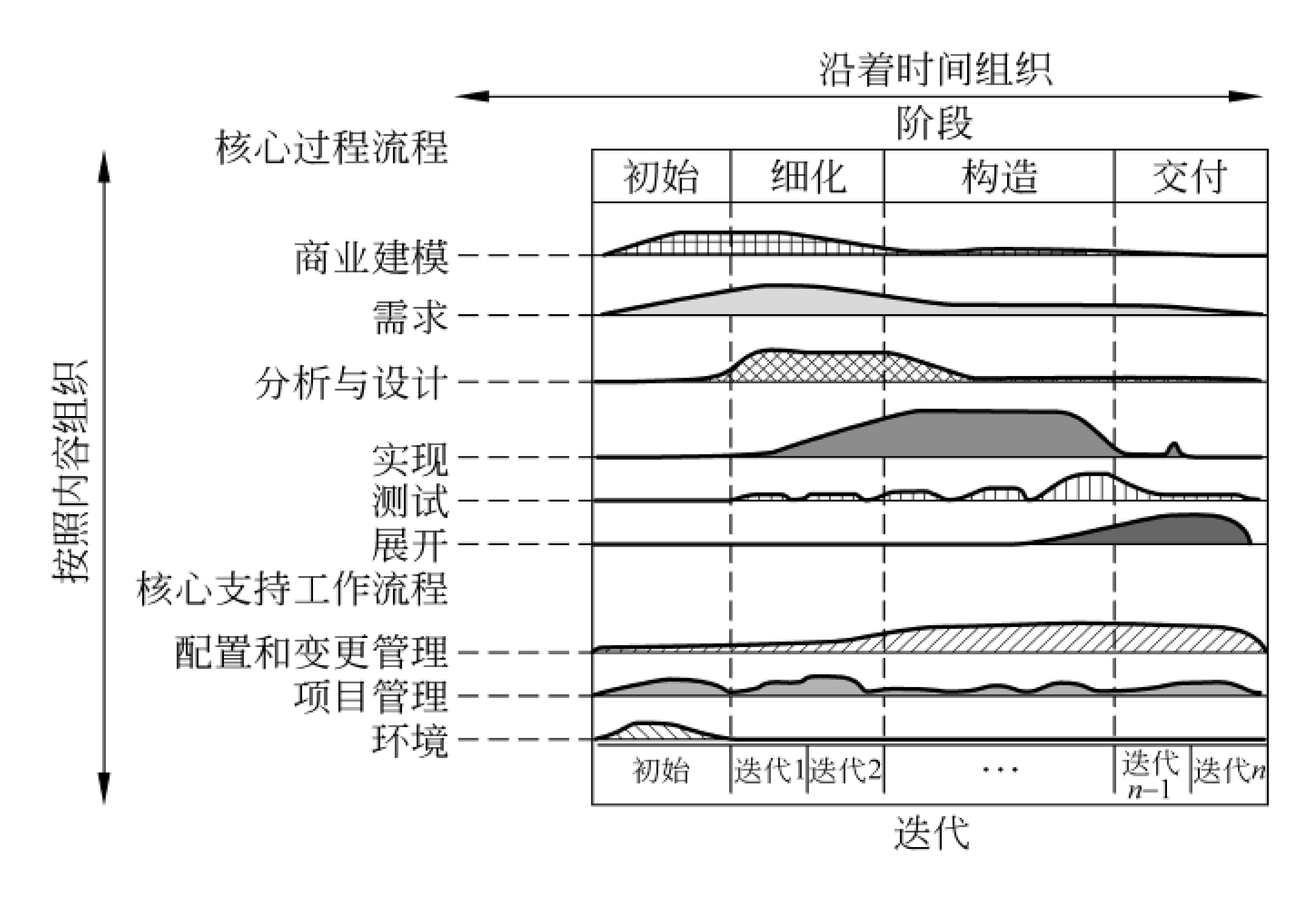


图 2-3 RUP 生命周期

过程随着时间动态组织，把软件的生存期划分为一些周期，每个周期都影响新一代产品。RUP 把每个周期划分为四个连续的阶段，每个阶段又细分为若干次迭代，每次迭代都要经过需求获取、分析、设计、实现和测试。

- 初始阶段(inception phase)；
- 细化阶段(elaboration phase)；
- 构造阶段(construction phase)；
- 交付阶段(transition phase)。

每个阶段的结果都是一个里程碑。里程碑是一个时间点，在这个时间点上必须做出重要的决策，达到一些关键的目标。

每个阶段都有明确的目标。

2.4.1 初始阶段

初始阶段确定所设立的项目是否可行，并确定系统的目标。初始阶段包括：

- 对需求有一个大概的了解，确定系统中的大多数角色和用例，但此时的用例是简要的，这些用例构成一个简要的用例模型。对给出的系统体系结构的概貌，细化到主要子系统即可。
- 考虑时间、经费、技术、项目规模和效益等因素。
- 识别并降低影响项目可行性的最不利的风险。
- 关注于业务情况，建立初始业务用例。

初始阶段的主要成果是：

- 前景文档(对核心项目要求、关键性质、主要限制的一般性的前景说明)；
- 初始的用例模型(完成 10%～20%)；
- 初始的项目术语表；

- 初始的商业用例,包括商业环境、验收规范以及成本预测;
- 初始的风险评估;
- 项目规划(其中明确阶段和迭代);
- 商业模型(根据需要可选);
- 一个或多个原型。

初始阶段结束时是第一个里程碑——生命周期目标里程碑。对初始阶段进行评估的准则是:

- 风险承担人对项目范围定义和成本/进度估计达成共识;
- 需求由主要的用例无二义地表达出来;
- 成本/进度估计、优先级、风险和开发过程的可信度;
- 开发出来的体系结构原型的深度和广度;
- 实际支出与计划支出的比较。

如果项目没有通过这个里程碑,就应该考虑取消或者重新思考。

2.4.2 细化阶段

细化阶段的目标是分析问题领域,建立一个健全的体系结构基础,编制项目规划,淘汰项目中风险最高的元素。为了达到这些目标,必须对系统有一个广泛的认识。体系结构的决策必须建立在对整个系统有一个理解的基础上,包括系统的范围、主要功能和非功能需求等。

细化阶段是四个阶段中最关键的。在这个阶段的最后,就认为已经完成最困难的工程,可以进行项目的结算,决定是否把它提交到构造和交付阶段。对大多数项目来说,这也相当于从低风险的运作到高成本、高风险运作的过渡。尽管过程必须能够适应不断的变化,但是细化阶段的活动必须保证体系结构、需求和规划有足够的稳定性,充分降低风险,从而才能够预算出整个开发过程的成本和进度。在细化阶段,根据项目的范围、大小和创新性,可能在一个或多次迭代中,建立一个可执行的体系结构。这一工作至少要解决初始阶段中找出的最重要的用例,而它们通常也揭示了项目的主要技术风险。虽然目标是为产品质量组件建立一个不断演化的原型,但也不排除开发一个或多个抛弃型的原型,来降低某些风险,或给投资人、客户及最终用户演示。细化阶段包括:

- 识别出剩余的大多数用例。对当前迭代的每个用例进行细化,分析用例的处理流程、状态细节以及可能发生的状态改变。细化流程时,可以使用程序框图和协作图,还可以使用活动图和类图分析用例。
- 降低重要的风险。
- 进行高层的分析和设计并做出结构性决策。所产生的体系结构基线包括用例列表、领域概念模型和技术平台。以后的阶段对细化阶段建立的体系结构不能进行大的变动。体系结构基线的稳定是细化阶段结束的准则。
- 为构造阶段制订计划。细化阶段完成,意味着能给出项目的成本和进度的估算。

细化阶段的成果是:

- 用例模型(至少完成80%,识别出了所有的用例和角色,以及大多数用例的描述);
- 一些增加的需求,包括非功能性需求以及任何与特定用例无关的需求;
- 软件体系结构描述;
- 可执行的体系结构原型;
- 修订后的风险表和商业用例;
- 整个项目的开发计划,包括粗略项目规划,显示迭代过程以及相应的评估准则;
- 更新的开发用例,指定要使用的过程;
- 初步的用户手册(可选)。

细化阶段结束时是第二个重要的里程碑——生命周期体系结构里程碑。此时检查系统详细的目标和范围,体系结构的选择以及对主要风险的解决。细化阶段的评估准则包括对以下问题的回答:

- 产品的前景是否稳定?
- 体系结构是否稳定?
- 可执行的演示是否强调了主要的风险元素,并且已经解决?
- 构造阶段的规划是否已经足够详细和准确,是否有可信的评估支持?
- 如果用当前的规划来开发整个系统,并且使用当前的体系结构的话,是否所有的风险承担人对当前的前景都达成一致?
- 是否实际的资源支出与计划的支出都是可接受的?

如果项目不能通过这个里程碑,则将取消或重新考虑。

阅读材料

在细化阶段末期,得到的系统构架为体系结构基线(Architecture Baseline),它是系统的"骨架",是开发人员目前和将来进行开发时都要遵循的标准。它包括早期版本的用例模型、分析模型、设计模型、部署模型、实型模型和测试模型(其中用例模型和分析模型较为成熟),这些模型中含有系统部件、中间件、要复用的遗产系统以及系统分布情况等。该基线与最终系统(对客户发布的产品)有同样的骨架。从最初体系结构基线到最终系统实现之间要经过几个内部发布,最后形成的体系结构基线为一个模型集合和一个体系结构描述,其中包括重要的用例及其实现。

建立可执行的体系结构基线是细化阶段的一个目标,进入构造阶段的体系结构基线应是坚实的。细化阶段结束时的体系结构基线在构造阶段变化不大,即体系结构基线到最终系统的体系结构之间的改动不大。

2.4.3 构造阶段

构造阶段主要是编码、单元测试工作,是人工最密集的阶段。这个时候,虽然允许有小的需求加入进来,但是应该尽量避免大的需求变动。目标是开发整个系统,并确保产品可以向用户移交。在构造阶段,将开发所有剩余的组件和应用部件,对它们进行测试,并集成到产品中。从某种意义上说,构造阶段是一个制造过程,包括:

- 识别出剩余的用例;
- 此阶段的每一次迭代开发都针对用例进行分析、设计、编码、测试和集成,所得到

产品满足项目需求的一个子集；

- 在代码完成后，要保证其符合某些标准和设计规则，并要进行质量检查。

构造阶段的产品是一个可以立即提交给最终用户使用的产品，它至少应该包括：

- 在特定平台上集成的软件产品；
- 用户手册；
- 对当前版本的描述。

构造阶段结束是第三个里程碑——初始运行能力。此时要决定软件、节点和用户是否已经准备好运行，并且项目没有出现任何高风险问题。这个版本通常叫做 α 版本。

关于α 版本的解释

事实上，软件开发人员不可能完全预见用户实际使用程序的情况。例如，用户可能错误地理解命令，或提供一些奇怪的数据组合，也有可能对设计者自认明晰的输出信息迷惑不解，等等。因此，软件是否真正满足最终用户的要求，应由用户进行一系列“验收测试”。一般采用称为 α、β 测试的过程，以期发现那些似乎只有最终用户才能发现的问题。

① α 测试是指软件开发公司组织内部人员模拟各类用户对即将面市软件产品（称为 α 版本）进行测试，试图发现错误并修正。测试不能由程序员或测试员全部完成。α 测试的关键在于尽可能逼真地模拟实际运行环境和用户对软件产品的操作，并尽最大努力涵盖所有可能的用户操作方式。经过 α 测试调整的软件产品称为 β 版本。

② β 测试是指软件开发公司组织各方面的典型用户，在实际工作中使用 β 版本，并要求用户报告异常情况、提出批评意见，然后软件开发公司再对 β 版本进行修改和完善。这种测试一般由最终用户或其他人员完成，不能由程序员或测试员执行。

构造阶段的评估准则包括对以下问题的回答：

- 这个产品版本是否足够稳定和成熟，可以在用户群中发布吗？
- 是否所有的风险承担人都已经准备好向用户提交？
- 实际的资源支出和计划的支出的比值是否可接受？

如果项目没有达到这个里程碑，必须推迟发布。

2.4.4 移交阶段

移交阶段的目的是把软件产品交付给用户群，包括相关的培训等内容，即：

- 进行最后的验收测试，完成最后的软件产品；
- 完成用户文档以及对用户培训等工作。

一旦产品提交给最终用户，通常会产生新的要求，如继续开发新版本，修正一些问题，或者完成某些被推迟的功能部件。当基线足够成熟，能够向最终用户领域发布时，就进入了交付阶段。这通常需要系统的一些可以使用的子集已经达到一定的质量要求，并且有用户文档，从而使交付产生积极的效果，包括：

- β 测试确认新系统达到用户的预期；
- 与被取代的旧系统并行操作；
- 功能性数据库的转换；
- 用户和维护人员培训；

- 向市场、分销商和销售人员进行新产品的展示。

移交阶段侧重向用户提交软件的活动，这个阶段包括几个典型的迭代，如 β 版本、通用版本的完成，以及对用户的反馈作出响应等，都需要大量的精力。但是，在生命周期的这一点上，用户反馈可能主要限于产品调整、配置、安装和易用性问题上。交付阶段的主要目标包括：

- 使用户可以自我帮助；
- 使风险承担人合作，使展开基线完整，并与前景评估准则一致。

这一阶段根据产品的不同，可以非常简单，也可以极其复杂。

交付阶段的终点是第四个重要的项目里程碑——产品发布里程碑。在这个点上，要确定是否已经达到目标，能否开始另一个开发周期。交付阶段主要的评估准则包括对以下问题的回答：

- 用户是否满意？
- 是否能够接受实际的和计划的资源支出的比值？

注意：RUP 中的每个阶段都可以进一步分解为若干次迭代。迭代是一个完整的开发循环，它的结果是可执行产品的一个版本，是正在开发的最终产品的一个子集，从一次迭代到另一次迭代，不断递增地成长，直到最后成为最终系统。

可以看出 RUP 虽然是基于迭代式开发，但是在整体的四个阶段划分上还是类似于瀑布式开发的软件过程。

2.5 RUP 过程的静态结构

2.5.1 软件过程元模型

RUP 是根据软件开发过程的元模型设计出来的方法，因此必须先对软件过程元模型进行了解。模型从某一角度出发，抽象出事物最重要的方面，并忽略或简化其他方面。软件过程也可以用模型来描述它的领域知识，例如图 2-4 的模型描述了 RUP 中测试规程的测试分析人员以及负责的工作。

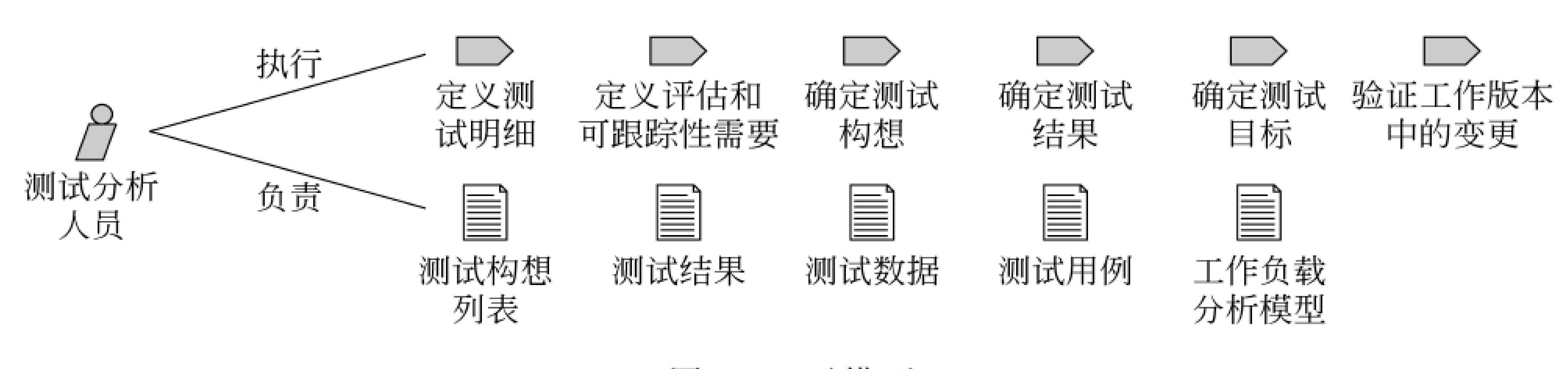


图 2-4 元模型

软件过程元模型显示了任何软件过程都是必不可少的角色(role)、活动(activity)、工件(work product)的概念。角色是对个人或作为开发团队的一组人职责的规定：具体个

人和角色的关系好比人和帽子的关系。活动就是角色执行的工作单元，而工件是工作的成品或半成品，如图 2-5 所示。

角色的职责具体体现在他执行活动和负责的工件上。工件是由活动生产出来的，是活动的输出，例如制定《编码规范》。然而，活动本身也可能以工件为输入，活动可能要求使用工件。例如编码活动要参考《编码规范》。当然工件既是活动的输入又是它的输出——活动修改工件，例如修改《编码规范》。

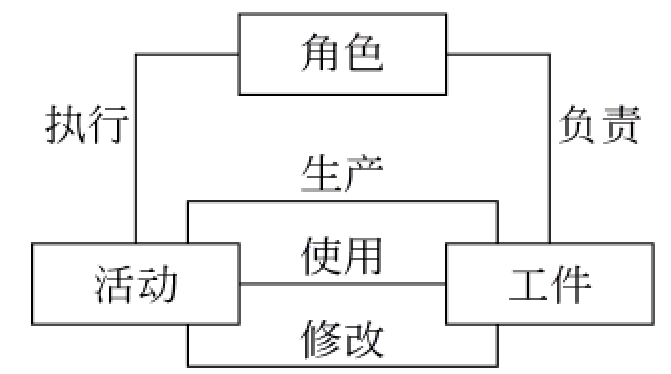


图 2-5　角色、活动、工件三者的关系

名词解释：角色、活动和工件

角色：描述个人或者一个小组的行为与职责。角色代表项目中个人承担的作用，并确定如何完成工作。RUP 预先定义了很多角色，包括分析员、开发人员、测试员、经理及其他。角色与个体(包括个人或小组)的关系为：某种角色对应的工作由一个个体完成，一个个体可以担任一种或多种角色。

活动(activity)：是一个有明确目的的独立工作单元。例如："系统分析员"角色完成的活动，包括制订需求管理计划、制订用例建模指南、获取常用词汇、获取涉众需求、确定前景、查找主角和用例、建立用例模型结构、管理依赖关系等。"架构设计师"角色完成的活动包括确定用例的优先级、架构分析、确定设计机制、确定设计元素、合并现有设计元素、说明分布、说明运行时架构、建立实施模型、制订设计指南、制订编程指南等。

工件：是活动生成、修改或使用的一段信息。工件是项目切实的成果，是项目为生产出最终的产品而制造或使用的东西。RUP 中的工件包括模型、模型元素、文档、源代码、可执行文件、工具等。角色执行某个活动前通常需要输入工件，活动结束后产生输出工件。

借用元模型，软件工程过程可定义谁在做什么、怎么做以及什么时候做，RUP 用四个主要的建模元素表达：

- 角色——谁；
- 活动——怎么做；
- 工件——做什么；
- 规程——什么时候做。

2.5.2　规程

仅把所有的工人、活动和产品都列举出来还不能够组成过程，另外还需要一种有效的方式，把产生有价值结果的活动序列描述出来，并显示工人之间的交互。规程是一个产生具有可观察的结果活动序列，规程也可以叫做工作流。在 UML 中可以用一个序列图、协作图或活动图来表示规程。

需要注意的是，规程中不能表示出活动之间所有的相关性，通常两个活动之间的相关性会比图中显示出来的更紧密，尤其是当它们涉及同一个工人或个人时，不能把规程解释

成由人机械执行的程序。

RUP 开发过程中有九种最基本的规程，称为核心规程，把工人和活动划分为不同的逻辑组合。核心规程被划分为六个核心过程规程和三个核心支持规程。

六个核心过程规程包括业务建模、需求、分析和设计、实现、测试、部署。

三个核心支持规程包括项目管理、配置和变更管理、环境。

尽管六个核心过程规程可能使人想起传统瀑布模型中的几个阶段，但应注意迭代过程中的阶段是完全不同的，这些规程在整个生命周期中一次又一次被访问。在一个项目实际的完整的规程中，九个核心规程在项目中轮流被使用，在每一次迭代中以不同的重点和强度重复。

1. 业务建模

大多数商业工程项目的主要问题在于软件工程和业务工程这两个领域无法进行交流，导致业务工程的输出无法正确地用作软件开发的输入，反之亦然。RUP 通过为这两个领域提供一个共同的语言和过程，同时说明如何创建和维护业务和软件模型之间直接的可跟踪性来解决这个问题。

在业务建模中，我们用所有的业务用例来为业务过程建立文档，这确保所有的风险承担人可以对机构到底需要支持什么样的业务过程达成共识。对业务用例的分析是为了理解业务到底是如何支持业务过程的，这些是用业务对象模型来建档的。

2. 需求

需求规程的目标是描述系统应该做什么，并且允许开发人员和客户就这个描述达成共识。主要是定义系统功能及用户界面，为项目预算及计划提供基础。为了达到这个目的，提取、组织、文档化需要的功能和约束，跟踪并记录所做的折中和决策。

3. 分析和设计

分析和设计规程的目标是说明在实现阶段是如何实现系统的。建立的系统要求是：

- 在特定环境下，完成用例描述中指定的任务和功能；
- 满足所有需求；
- 其构造确保它是健壮的(如果功能需求发生变化时，易于更改)。

分析与设计就是把需求分析结果转换为分析与设计模型，其结果为设计模型以及一个可选的分析模型。设计模型是源代码的一个抽象，也就是说，设计模型的作用是一个“蓝图”，描述了如何对源代码进行组建和编制。设计模型由设计类和一些描述组成，设计类被组织成具有良好接口的设计包和设计子系统，而描述则体现了类的对象如何协同工作实现用例的功能。

设计活动以体系结构为中心，构建和确认这个体系结构是早期迭代的重点。体系结构用结构视图来表示，这些视图表达了主要的结构设计决策。从本质上，体系结构视图是整个设计的抽象或简化，丢掉细节，使重要的特征明显地表现出来。体系结构对于开发好的设计模型以及对于提高系统开发过程中任何模型的质量而言，都是重要的载体。

4．实现

实现是为了把设计模型转换为实现结果，并做单元测试，集成为可执行系统。其目的是：

- 通过分层次地组织实现子系统来定义代码的结构；
- 用组件（源文件、二进制文件、可执行文件等）的形式来实现类；
- 对所开发的组件进行单元测试和集成测试；
- 把每个实现人员（团队）的工作成果集成到一个可执行的系统中。

系统是通过实现组件实现的。RUP 描述了如何重用已有组件或实现新的组件，使系统更易于维护，提高可重用性。组件被构造成实施子系统。子系统以带有附加结构或管理信息的目录形式表示。

5．测试

测试是为了验证所有需求是否已经被正确实现，对软件质量提出改进意见。其目的是：

- 验证对象之间的交互；
- 验证是否恰当地集成了软件的所有组件；
- 验证是否所有需求被正确实现；
- 在发布软件之前，找到错误并改正。

RUP 是用迭代的方法，意味着在整个项目中都在进行测试，从而尽早地发现错误，从根本上降低修改错误的成本。测试分别从可靠性、功能性、应用性能和系统性能四个方面进行。对每个方面都描述了如何经历规划、设计、实现、执行和评估的测试生命周期。另外，描述了何时及如何引入测试自动化的策略。测试自动化对迭代方法尤其重要，在每次迭代结束和产品新版本开始时都要进行回归测试。

6．部署

部署的目的是为了成功地开发出产品版本，并把软件提交给它的最终用户。包括一系列的活动：

- 制作软件的外部版本；
- 软件打包；
- 分发软件；
- 为用户提供帮助和支持；
- 培训用户及销售人员。

在许多情况下还包括：

- 规划和实施 β 测试；
- 移植现有软件或数据；
- 正式接受。

尽管部署活动大部分集中在交付阶段，但许多活动还需要在较早阶段就进行准备。

7. 项目管理

项目管理的工作是为软件开发项目提供计划、人员分配、执行、监控等方面指导，为风险管理提供框架。它是平衡竞争目标、管理风险和战胜困难、成功地提交一个满足客户和用户需要产品的艺术。这一规程侧重于一次迭代开发过程的特定内容，目标是：

- 为管理软件密集型项目提供框架；
- 为项目的规划、人员管理、运行和监督提供实用的指南；
- 提供一个管理风险的框架。

这个规程并不是成功的秘诀，它只是提供了一个能够有效提高项目管理质量的项目管理方法。

8. 配置和变更管理

在配置和变更管理这个规程中，描述了如何控制由共同完成同一项目的许多人制造出来的大量产品。主要工作是跟踪并维护系统开发过程中产生的所有制品的完整性和一致性。控制有助于避免混乱，保证各个产品不会因为以下问题产生冲突：

- 同时更新——当一个或多个工人单独地做同一个产品时，最后一个人可能会破坏前面人的工作。
- 有限的通知——当多个人共享的几个产品中的错误被更正时，可能有些人没有得到有关修改的通知。
- 多个版本——大多数大程序都是以版本不断进化的形式开发的。一个版本可能提供给用户使用，另一个用来测试，而第三个用来继续开发。如果其中任何一个版本发现问题，就需要把修改的信息传播给其他版本。

在这个规程中，将尽可能对重复工作、无效的变化进行控制和监视，以避免由此产生的混乱。它主要提供一些管理多个软件版本的指南，跟踪开发版本，保证按照用户定义的规格说明进行开发，强制采取节点专用开发政策。它对如何并行开发、在多节点上开发以及使建造过程自动化进行了描述。这些在迭代开发过程中尤其重要。另外，该规程还描述了如何进行审核跟踪，把谁、什么时候、对什么产品做的什么修改记录下来。此外，它还包括变更需求管理，也就是如何报告和管理故障，以及如何使用故障数据来跟踪进展和发展倾向。

9. 环境

环境规程的目的是为软件开发组织提供软件开发环境——提供开发团队需要的过程和工具支持。该规程的重点是在项目环境中配置过程的活动。另外就是描述如何在一个机构内实现过程。环境规程还包含一个开发工具包，提供一些定制过程的指南、模板和工具。

2.6 RUP 中的最佳软件实践

从另一个角度理解,RUP 可认为是最佳软件开发经验的总结,它包括了软件开发中的六大经验,即迭代式开发、管理需求、使用基于组件的软件体系结构、可视化建模、软件质量保证、控制软件变更,它们是判断是否真正实施 RUP 的一个重要标准。RUP 的核心是为软件开发团队提供指南、文档模板和工具,以使整个团队能够最有效地利用这些最佳的软件开发经验。

2.6.1 迭代式开发

当今的软件系统十分复杂,很难按照首先定义整个问题、设计整个系统、构建软件、最后测试产品的顺序线性进行。在软件开发的早期阶段就想完全、准确地捕获用户的需求几乎是不可能的。实际上,我们经常遇到的问题是需求在整个软件开发工程中经常改变。由此需要一种迭代的方法,通过不断地细化来增进对问题的理解,在多次迭代的基础上递增地得到一个有效的解决方案。迭代式开发允许在每次迭代过程中需求可能有变化,通过不断细化来加深对问题的理解。RUP 支持迭代开发,在生命周期的每个阶段都注重风险最高的问题,显著地降低项目的风险系数。这种迭代的方法通过可见的进展情况、可执行版本来促进最终用户的参与和反馈,从而有助于降低开发过程中的风险。而且,因为每次迭代结束时都提交一个可执行的系统版本,使开发团队能够始终将注意力放在产品上,经常性的阶段检查也确保项目按计划进行。迭代方法还很容易容纳需求、特色或规划上的改变。迭代式开发不仅可以降低项目的风险,而且每次迭代过程以可以执行版本结束,可以鼓舞开发人员。如图 2-6 所示,RUP 中的每一次迭代都要经过这些规程。

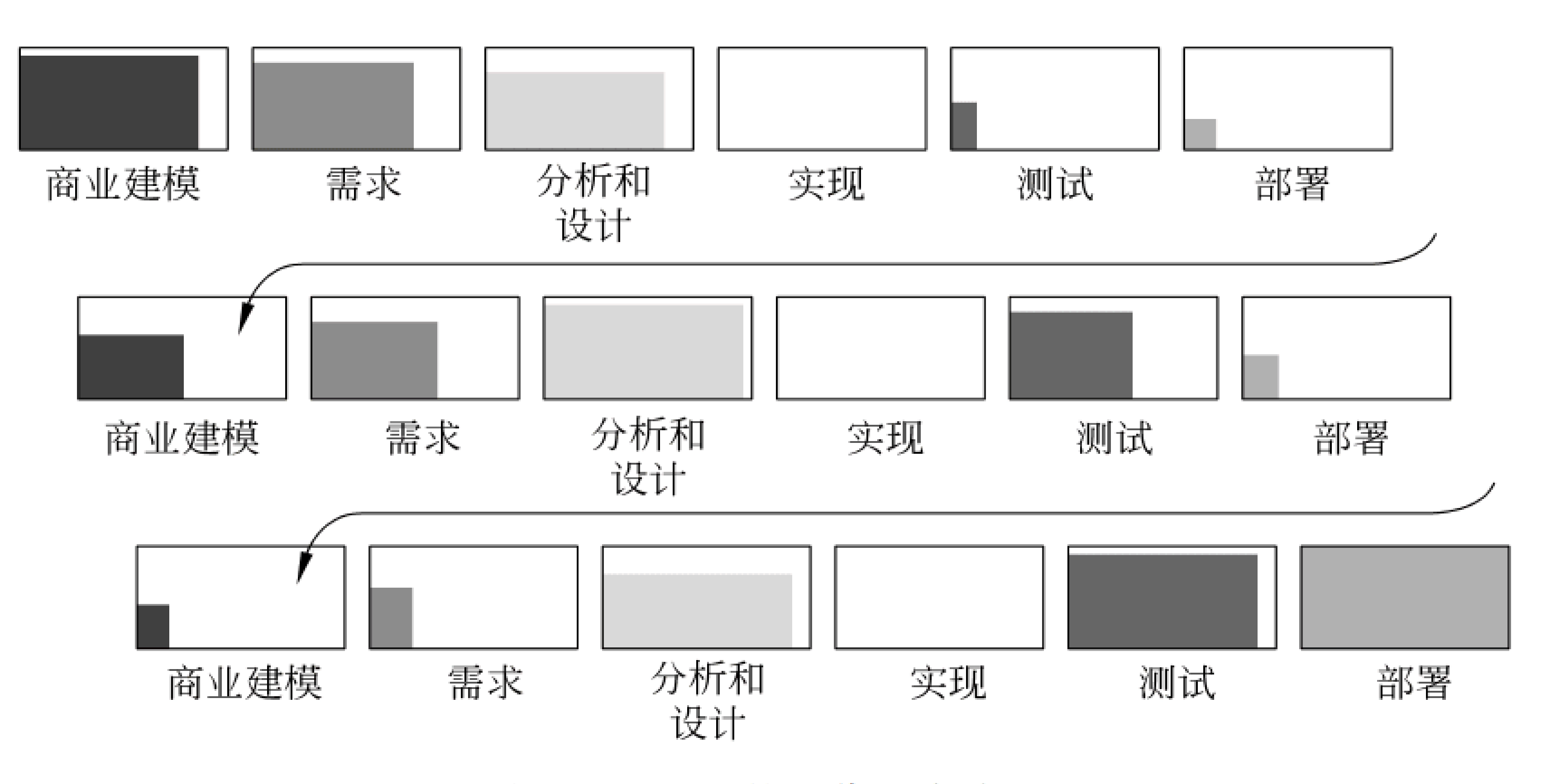


图 2-6　RUP 的迭代开发流程

举个例子,读者可能会认为电影制作是一个直进的过程,首先写剧本,然后解决如何将文字表现为动作,拍摄电影并剪辑,最后观看效果。这可能是一个传统的瀑布式方法。

但正如这种瀑布式方法在编写软件时是失败的一样，在电影制作时这种方法也会失败。

电影拍摄的过程是更加迭代性的。演员的工作从剧本开始，由这种交互开始修改。例如，轰动一时的《指环王》，它的剧本基于 J. R. R. Tolkien 的名著，几乎每天都在和演员交互后进行修改。导演 Peter Jackson 把这一创新性的过程描述为控制下的混沌。在电影结束之前，剧本被修改了很多次。每一次修改（或迭代）都产生了一个更多的（也是更好的）版本，更加接近最终的版本。

2.6.2 管理需求

RUP 把需求定义为"（正在构建的）系统必须符合的条件或具备的功能"。电气和电子工程师学会（IEEE）使用的定义与此类似。

先看看以下例子：

- Standish Group 从 1994 年到 1997 年的 CHAOS 报告证实，导致项目失败的最重要的原因与需求有关，进一步证实了与成功项目关系最大的因素是良好的需求管理。
- 1997 年 12 月，Computer Industry Daily 报道了 Sequent Computer Systems 公司的一项研究，该公司对美国和英国 500 名 IT 经理作调查后发现，76%的受访者在他们的事业中经历过完全的项目失败。其中提到最多的导致项目失败的原因就是"变更用户需求"。

这充分说明了管理需求的必要性，它可以提高项目的成功率。软件开发的任务就是保证开发出来的软件符合用户的需要，管理需求的目标是剔去不符合用户需要的需求，以降低软件开发的风险。因为有时候用户对开发中的产品仅有一个模糊的想法，这时候需求可能不是很重要或简单的，而且这些需求可能还会随时间而变化。这样一个初始设定好的需求似乎并不是最好的需求，RUP 提出一种主动的需求管理策略，可以迭代地改进当前的需求，满足用户需要。

阅读材料

有几种原因使需求分析变得困难：客户说不清楚需求；需求自身经常变动；分析人员或客户理解有误。

（1）客户说不清楚需求

有些客户对需求只有朦胧的感觉，说不清楚具体的需求。例如全国各地有很多政府机构搞网络建设，这些单位的领导和办公人员大多不清楚计算机网络的具体作用和需求，只好要软件系统分析人员替他们设想需求。既然系统分析人员一人说了算，有的便可以为所欲为，需求变得不实在。

有些客户心里非常清楚想要什么，但却说不明白。读者可能不以为然。就举日常生活的事例吧，例如说买鞋子。我们非常了解自己的脚，但没法说清楚脚的大小和形状。只能拿鞋子去试，试穿时感觉到舒服才会买鞋。

如果客户本身就懂软件开发，能把需求说得清清楚楚，这样的需求分析将会非常轻松、愉快。如果客户不懂软件，但信任软件开发方，这事也好办。分析人员可以引导客户，

先阐述常规的需求，再由客户否定不需要的，最终确定客户真正的需求。最怕的就是“不懂装懂”或者“半懂充内行”的客户，他们会提出不切实际的需求。

(2) 需求自身经常变动

唐僧曾说：“妖要是有了仁慈之心，就不再是妖，是人妖。”(《大话西游之大圣娶亲》)

连妖都会变心，别说人了。所以喜新厌旧乃人之常情，世界也因此变得多姿多彩。

软件的需求会变化吗？

让我们先接受“需求会变动”这个事实吧，免得在需求变动时惊慌失措。明白“需求会变动”这个道理后，在进行需求分析时就要留点神：

① 尽可能地分析清楚哪些是稳定的需求，哪些是易变的需求。以便在进行系统设计时，将软件的核心建筑在稳定的需求上，否则将会吃尽苦头。

② 在合同中一定要说清楚“做什么”和“不做什么”。如果合同含含糊糊，日后扯皮的事情就多。要防止像抗战时期山东省主席韩复渠那样，在别人请他喝酒吃饭时他什么都点头(人家就更加献殷勤)，吃完了他就宣布刚才答应的事都不算数，便扬长而去。

(3) 分析人员或客户理解有误

有个外星人间谍潜伏到地球刺探情报，它给上司写了一份报告：“主宰地球的是车。它们喝汽油，靠四个轮子滚动前进。嗓门极大，在夜里双眼能射出强光。……有趣的是，车里住着一种叫作‘人’的寄生虫，这些寄生虫完全控制了车。”

这是典型的理解有误。软件系统分析人员不可能都是全才。客户表达的需求，不同的分析人员可能有不同的理解。如果分析人员理解错了，可能会导致开发人员白干活，吃力不讨好。所以分析人员写好需求说明书后，要请客户方的各个代表验证。如果问题很复杂，双方都不太明白，就有必要请开发人员快速构造软件的原型，双方再次论证需求说明书是否正确。

由于客户大多不懂软件，他们可能觉得软件是万能的，会提出一些无法实现的需求。有时客户还会把软件系统分析人员的建议或答复给想“歪”了。

有一个软件人员滔滔不绝地向客户讲解在“信息高速公路上做广告”的种种好处，客户听得津津有味。最后，心动的客户对软件人员说：“好得很，就让我们马上行动起来吧。请您决定广告牌的尺寸和放在哪条高速公路上，我立即派人去做”。

RUP在软件的开发周期内进行需求管理，这意味着项目需求会被反复和逐渐地确定、证明、评估和改进。为了进行需求管理，RUP描述了如何提取、组织系统的功能和约束条件并将其文档化，如何跟踪和建档折中方案和决策，并且易于表达商业需求和交流。

用例和脚本的使用已被证明是捕获功能性需求以及使最终系统更充分地满足用户需要的有效方法。功能性的需求用术语“用例”来描述。软件系统中非常重要的非功能性的需求，例如对系统有很大影响的系统特性，包括系统易用性、可靠性、系统性能和可支持性等，这些需求不能用用例来描述，也应该被确定和管理，例如网站访问响应时间应小于0.01秒等。

2.6.3 基于组件的体系结构

组件是一个具有清晰功能的可替换的软件模块。RUP 鼓励使用组件来组合成一个系统。基于组件的开发有以下的优点：它使软件在本系统和其他系统中的重新使用上更加方便，它在系统设计中提供了方便的抽象概念，并且允许高效的并行开发。

基于独立的、可替换的、模块化组件的体系结构更加容易升级和维护，有助于管理复杂性，提高重用率。RUP 侧重在利用资源进行规模开发之前，注重体系结构早期的开发和基线。RUP 描述了如何设计一个有弹性的、能适应变化的、易于理解的、有助于重用的软件体系结构，提供了一个系统的方法用新的和已有的组件定义体系结构。

2.6.4 可视化建模

前面已提到过模型，模型是现实世界实体或过程的简化表达，它帮助我们想象和勾画出现实世界的形象。可视化建模（visual modeling）就是利用围绕现实想法组织模型的一种思考问题的方法。模型对于了解问题、与项目相关的每个人（客户、行业专家、分析师、设计者等）沟通、模仿企业流程、准备文档、设计程序和数据库来说都是有用的。建模促进了对需求的更好的理解、更清晰的设计、更加容易维护的系统。模型通过过滤非本质的细节信息，成为描述复杂问题的抽象（abstraction），它使问题更容易理解了。

抽象是一种允许我们处理复杂问题的基本能力。千百年以来，工程师、艺术家和工匠一直在实施某项工程之前，先建立模型提炼出它的设计方案。软件系统的开发也并不例外。为了建立复杂的系统，开发者必须抽象出系统的不同的视图，使用精确的符号建立模型，验证这些模型是否满足系统的需求，并逐渐添加细节信息把这些模型转变为实现（implementation）。

模型的一个例子就是原型，它可以用于模拟、测试或设计方面的实际工作。原型用于生成能够说明系统某方面或某种特性的信息。RUP 鼓励架构设计者通过创建原型来证明他们的架构概念。在 RUP 中还有其他类别的模型，例如用例图等，这些模型可以用来减小设计风险。一般来说在系统模型中比在实物中更有利于发现错误和缺点，因此建立这些模型是解决错误的最廉价方式。

建模有助于我们用简化的方法去理解复杂的系统，建立复杂系统的模型是因为我们没法理解整个系统。人类理解复杂性的能力是有限的。这个观念可以在世界上的建筑中看到。如果希望在后院中建立小屋，可以立即开始建造；如果希望建新房子，甚至建摩天大楼，就绝对需要先设计一张蓝图了。在软件的世界中这也是一样的。由源代码开发环境，例如 Visual Basic 中设计的窗体，为程序员提供的全局视图是很难全面描述开发项目的。构造模型允许设计师把全部精力放在考虑项目中的组成部分如何交互的全局情况，而不会陷入每个组成部分的具体细节信息中。

RUP 往往和 UML 联系在一起，利用 UML 对软件系统建立可视化模型，以帮助人们管理软件的复杂性。RUP 告诉我们如何可视化地为软件建模，以用来表达体系结构、组

件的结构和行为。这样就可以隐藏细节,使用图形化的基本模块来写代码。使用抽象可视化有助于进行不同方面的沟通,显示出系统元素是如何组织在一起的;保证各部件与代码一致;维护设计和实现的一致性;促进无二义性的交流。UML 正是成功进行可视化建模的基础。

2.6.5 软件质量保证

RUP 分为四个阶段,每一个阶段的重点放在开发周期内一个特定的方面:起始、精化、构建和产品化。在每一个阶段中 RUP 的最佳实践给我们机会验证开发中项目的进度和质量,从而尽可能早地找到错误和潜在的改进。

概括地说,在 RUP 中每一次迭代都要检验上一次增加的质量。首先检验我们是否已经理解了问题,并且存在一个可靠的业务场景。这里场景用来描述流经用例的路径,从用例开始到结束遍历这条路径上所有基本流和备选流。因为现在的软件几乎都是用事件触发来控制流程的,而同一事件不同的触发顺序和处理结果就形成了事件流。这种在软件设计方面的思想也可引入到软件测试中,可以比较生动地描绘出事件触发时的情景,有利于测试设计者设计测试用例,同时使测试用例更容易理解和执行。

名词解释

场景法是 RUP 所提倡的测试用例设计思想。现在的软件大部分是由事件触发来控制流程的,事件触发时的情景就是所谓的场景。在测试用例设计过程中,通过描述事件触发时的情景,可以有效地激发测试人员的设计思维,同时对测试用例的理解和执行也有很大的帮助。

如果需求规格说明书是采用 UML 的用例设计方式进行的话,测试人员可以比较轻松地通过把系统用例影射成测试用例的方法来设计测试用例。需要覆盖系统用例中的主场景和扩展场景,并且适当地补充各种正反面的测试用例以及考虑出现异常的情形。

如果没有理解问题,或者不存在可靠的业务场景,则停止这个项目以节省时间和金钱,或者继续努力直至达到这个目标。在进行迭代之前,同时也检验是否已经建立了一个十分牢固的架构。持续的质量验证可以使这样做变得容易。

软件性能和可靠性低下是影响软件使用的最重要的因素。因此应该根据基于可靠性、功能、应用性能和系统性能的需求对软件的质量进行评估。RUP 可以帮助进行这些类型的规划、设计、实现、执行和评估。在 RUP 中软件质量评估不再是事后进行或单独小组进行的各自活动,而是内建于过程中的所有活动,让所有的人员都参与,使用目标准则,这样可以更早地发现软件缺陷。

2.6.6 控制软件变更

需求总是随着时间不可避免的变化,由于许多不同的原因,例如项目的投资者改变了主意,开发者对需求有了更深的认识,设计产生的效果明显等。这个时候,如何有效地控

制软件需求变更就成了问题。有项调查显示瀑布式软件开发经常失败的一个原因就是没有很好地处理变更。但 RUP 里提出的迭代和增量式方法可以很好地控制软件变更，使连续的变更更加容易，通过迭代得到的解决方案质量更好。

迭代式开发中如果没有严格的控制和协调，整个软件开发过程很快就陷入混乱之中，RUP 描述了如何控制、跟踪、监控、修改变更，从而保证迭代开发过程的成功。它也指导人们如何通过控制所有对软件制品（如模型、代码、文档等）的变更，从而隔离来自其他工作空间的变更，以此为每个开发人员建立安全的工作空间。它还描述了如何自动化集成和建立管理，从而使一个团队的工作团结一致。

2.7　RUP 中的关键原则

在整个 IT 研发思想方面，IBM 公司倡导业务驱动开发（business-driven development，BDD）的思想，以改变长期以来研发部门与业务部门以及运行维护部门的隔离，系统开发和运行效率低的问题。该思想认识到应该将开发项目集中于业务需求，而不是以 IT 为中心的解决方案，提出加强业务、研发和运作的沟通，统一透明地进行，从业务需求、软件研发到生产运作的全周期管理。业务驱动开发使组织能够控制软件和系统开发的业务流程。

基于这个思想，IBM 公司从成千上万的软件开发项目中，将 RUP 六个最佳实践演变为六个关键原则。这些原则描述了构建、部署和改进软件密集型系统的工业最佳实践的特征：

- 提高过程的适应性。
- 设定涉众优先级。
- 跨团队协作。
- 迭代地证明价值。
- 提高抽象层次。
- 持续关注质量。

下面将按顺序解释上述这些原则，强调的是：

- 通过应用原则得到的好处；
- 最能体现原则的行为模式；
- 公认的与原则相悖的“反模式”或行为（它们会危害软件开发项目）。

2.7.1　提高过程的适应性

好处：增强生命周期的效率，开放或坦诚的风险沟通。

模式：随着项目生命周期中不确定因素的消减，精确性和形式化程度不断加深。使过程适应于项目团队的规模和分布，适应应用的复杂性以及灵活性需要。不断改进过程。

反模式：精确地计划或估计，遵循静态计划管理方式。过程越多越好。在整个生命

周期中不分轻重地使用过程。

过程越多(例如使用更多工件,创建更详细的文档,开发和维护更多需要同步的模型以及实施更多正式的评审)并不代表越好。

第一,我们需要根据项目需要正确裁减过程的大小。随着项目规模越来越大,分布范围越来越广,使用的技术越来越复杂,项目涉众越来越多,严格符合法规章程的需求越来越迫切,过程也就需要越专业。但是,对于小型项目,团队在同一地点工作的,技术已知,过程就应该为轻量级,这些依赖关系如图 2-7 所示。

需要多大规模的流程?

简单升级
研究和开发原型
静态Web应用程序

动态Web应用
程序封装的应用程
序基于组件 (J2, .Net)

旧系统升级由多种
系统组成的系统实时、
嵌入式系统可证明的质量

什么时候适合较少的流程?
- 工作地点相同的团队
- 较小的、较简单的项目
- 项目干系人很少
- 早期生命周期阶段
- 内部强加的约束

什么时候适合较多的流程?
- 分布于不同地点的团队
- 大型项目
- 项目干系人很多
- 后期生命周期阶段
- 外部强加的约束
  - 标准
  - 合同需求
  - 法律需求

图 2-7　推动过程专业程度的因素

第二,项目的过程应适应生命周期的各个阶段。项目开始时,通常将面对许多不确定的因素,并且必须积极鼓励创造性地开发处理业务需求的应用程序。通常过程越多创造性越低(而非创造性越高),因此必须在每天都会遇到的不确定情况的项目早期使用最少的过程。另一方面,在项目后期,需要引入更多的控制(诸如变更控制委员会)来防止不希望出现的创造性开发和关联风险,它们会在后期将缺陷引入产品,而这又将产生更多的过程。

第三,企业应致力于不断改进过程。请考虑在每次迭代之后执行评估,并在项目最终总结经验教训,然后加以充分利用来改进流程。鼓励所有团队成员不断寻找改进的机会。

第四,使项目计划和关联的估计与项目的不确定性保持平衡至关重要。这意味着,在有着大量不确定性的项目早期,计划和关联估计的重点应放在全面性计划和估计,而不应将目标放在细节问题上(当时这些问题实际上并不存在)。早期开发活动的目标应去除不确定性,以逐步使计划更为准确。

确定项目规程的因素很多,包括项目规模、团队分布、技术复杂度、涉众数量、灵活性需求及项目生命周期中所处的位置。

这一原则的反模式是永远认为更多过程和更详细的前景计划是更好的。强制进行早期估计,并依赖于这些估计。

2.7.2 设定涉众优先级

好处：按业务和用户需要调整应用程序，减少定制开发，优化业务价值。

模式：定义并理解业务过程和用户需求；区分项目、需求与软件能力的优先次序；理解可以使用哪些资产；并平衡用户需求与资产复用。

反模式：在项目工作开始前实现精确和详细的需求。需求是达到客户解决方案的焦点。

这一原则阐述了平衡业务和涉众需要的重要性，两者往往是冲突的。涉众是 RUP 中的一个名词，表示软件开发中涉及的各种角色，如用户、设计人员、开发人员乃至测试人员等。大多数项目的涉众都倾向于让应用程序严格执行他们希望的操作，同时将应用程序开发成本和进度时间减到最少。但是，这些优先方面经常会冲突，如图 2-8 所示。例如如果将以前的应用程序打包，则可以得到一个较低价格、更快交付解决方案，但是必须提前考虑许多需求。相反，如果选择从头构建应用程序，则可以处理所有的需求，但是预算和项目完成日期就会超出限度。



图 2-8 平衡需求处理组件的使用

使用组件可大大降低交付一组特定功能的成本并减少其进度安排。在很多情况下，必须在某些功能或技术需求上作出妥协，诸如平台支持、性能或占用空间（应用程序的实际大小）。

要能够平衡需求，必须首先有效地管理需求。开发团队处理需求列表中的元素，必须了解业务和项目涉众的需要，并对其划分优先级。这说明要捕获业务流程并将其链接到项目和软件功能，以使我们可以有效地划分项目和需求的优先级，并按照对增加的应用程序和发展的项目涉众需要的理解来修改优先级。这还说明需要客户或客户代表参与到项目中，以确保了解他们的需要。

同时，还需要将开发活动侧重于项目涉众的需要。例如通过利用用例推动开发和以用户为中心的设计，开发流程在项目期间要能适应项目涉众需要的变化，从而用于更改业务以及进一步理解对于业务和最终用户真正需要的功能。

最后，需要知道哪些资产可用，并平衡资产重用和项目涉众需要。资产包括旧的应用程序、服务、可重用组件和模式等。在许多情况下，资产重用可降低项目成本。重用经证明的资产通常意味着新应用程序的质量更高。缺点在于，在许多情况下必须在资产重用和满足用户需要之间做出权衡。重用组件可降低一个功能 80% 的开发成本，但同时也只能处理 75% 的需求。因此，有效重用需要不断地平衡资产重用和项目涉众需要。

使用一个组件可以从根本上降低成本和提前发布一组特定功能。在很多情况下它可能还需要牺牲一些功能或技术需求，例如平台支持，性能或覆盖区（应用程序的大小）。

遵循这一原则的反模式是在项目开始详细记录精确的需求，强制涉众接受需求，然后

对需求的变更进行协商，而需求的每一点变更都将增加项目的成本或持续时间。由于一开始就锁定了需求，降低了使用已有资产的能力，于是迫使进行定制化的开发。另一个反模式是构建一个只满足最强势涉众的需求系统。

2.7.3 跨团队协作

好处：团队生产力，更好地结合商业需求与软件系统的开发和运作。

模式：激励团队成员做得最好。分析师、开发人员和测试人员的跨职能合作。管理不断演进的工件和任务，通过集成环境来加强对协作、进度和质量的了解。确保业务、开发和运作团队作为一个集成的整体高效地工作。

反模式：培养英雄个体并为他们配备强大的工具。

软件是由才华横溢、积极上进的人员通过紧密协作创造出的。许多复杂的系统要求一些具有不同技能的项目涉众进行协作，而大型项目通常会跨越地理和时间的界限，从而进一步增加了开发流程的复杂性。这就是为什么人员问题和协作（一些人称之为软件开发的“软”元素）会成为灵活开发团队的主要侧重点。遵循此原则要求回答以下问题，包括：

- 如何激励人们做得最好？
- 如何在同处一地与分散的软件团队中进行协作？
- 如何在负责业务、软件开发和 IT 操作的团队之间进行协作？

有效协作的第一步是激励团队中的个人做得最好。自我管理团队的概念已在项目团体中得到普及，它的基础是使团队承诺负责应交付的成果，然后由团队决定和判断影响交付成果的所有问题。当人们意识到需要真正负责最终结果时，会更主动地去保质完成工作。如 RUP 的灵活性声明所陈述的：“为受到激励的人员构建项目。为他们创造环境，支持他们的需要并相信他们能完成工作。”

第二步是鼓励跨职能协作。如 Walker Royce 所说的“软件开发是一项团队工作。”迭代方法更需要团队密切配合工作。我们要拆除分析人员、开发人员和测试员之间通常存在的壁垒，图 2-9 拓宽这些角色的职责以确保在快速变化的环境中进行有效协作。每个成员都需要了解任务和项目远景。

在团队发展到一定规模时，要提供有效的协作环境。这些环境使度量值收集和状态报告变得更为方便和自动化，并使围绕着配置管理的构建管理和日志自动化。这种效率可减少会议次数，使团队成员将更多时间用在具有更高生产率和创造性的活动上。这些环境还应通过简化交流、使处于不同地区和时区的团队成员能够沟通，以此达到更有效的协作。这些环境的示例包括从共享项目房间到联网或基于 Web 的解决方案，例如集成开发环境、配置和变更管理环境。

随着软件对于核心业务流程越重要，对团队间有效协作的需求也随之增加。在大多数组织中，负责运行业务、开发应用程序和运行应用程序的团队之间缺乏交流和沟通。

此原则下最终目标是围绕业务、软件和操作团队集成协作。由于软件对于如何运行业务变得越来越重要，因此需要在决定如何维持当前和将来业务运转的团队、开发支持软

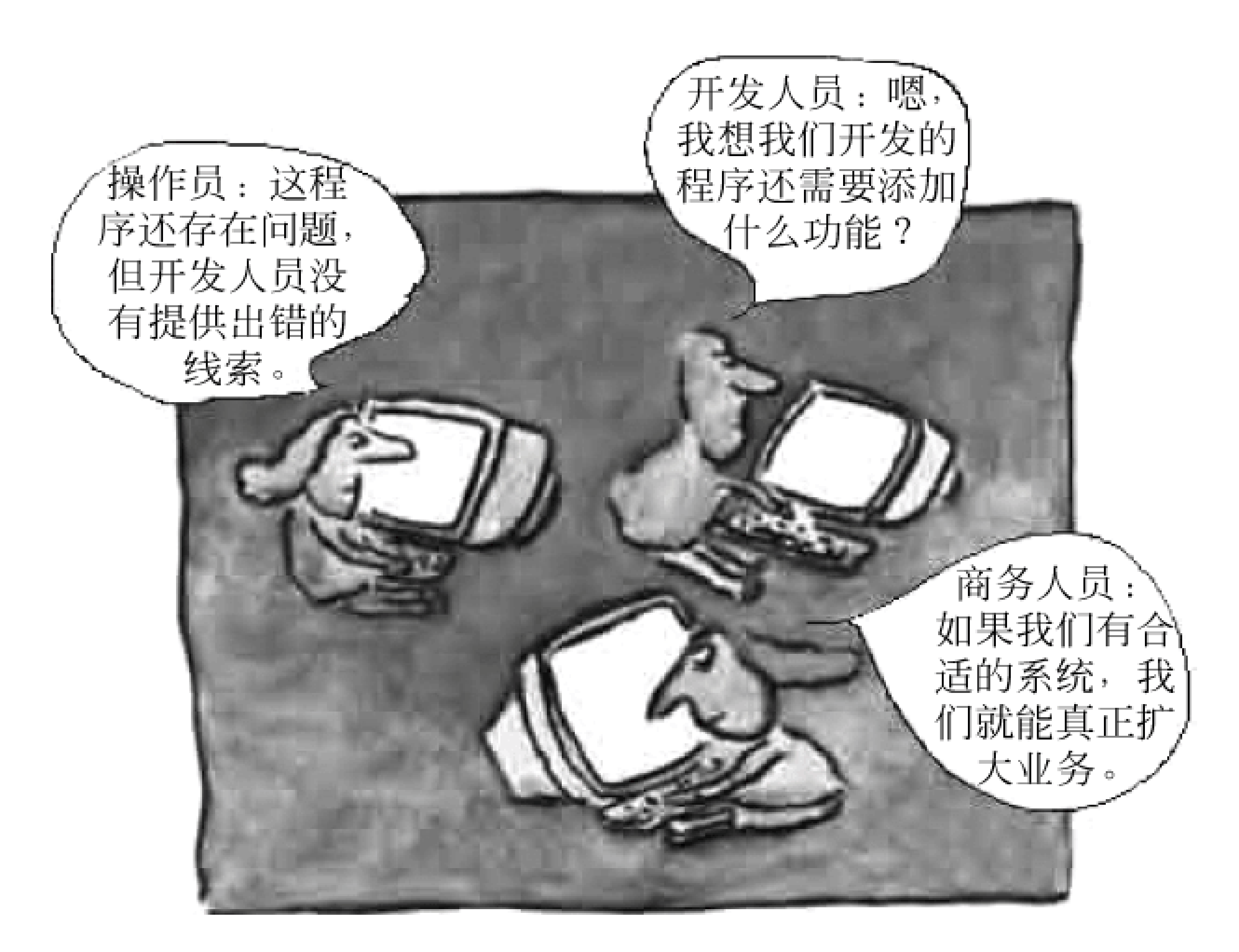


图 2-9　跨业务、开发和运作团队协作

件系统的团队和负责操作软件系统的团队之间建立紧密的协作。在大多数公司中，这三个组之间不能进行很好的沟通。

随着软件在如何运作业务中变得越来越重要，需要与负责如何运行业务，如何开发应用程序，以及如何运行应用程序的周围团队更紧密地合作。目前在一些公司中，这三个小组的交流都不通畅。

这一原则的反模式是培养英雄式的开发者，他们愿意超长时间工作，包括周末。一个相关的反模式是培养高度专业的人，并为他们的工作配备强大工具，他们与其他团队成员的合作很有限，而且不同工具间的集成也很有限。这里的假设是如果每个人都做好自己的工作，最终结果就会是好的。

2.7.4　迭代地证明价值

好处：及早降低风险，在项目整个过程中具有更高的可预测性，涉众之间相互信任。

模式：使用一次迭代过程进行自适应的管理。首先解决主要技术、业务和风险。通过在每次迭代发布递增的用户价值获得反馈。

反模式：详细计划整个生命周期，用计划记录变数。详细计划是更好的计划。通过复审规范来评估状态。

此原则下有如下几个需要。

第一个是必须交付递增值以启用早期和持续的反馈。可通过将项目分成一组迭代来完成此需要。在每次迭代中，执行应用程序的一些需求、设计、实施和测试，从而生成更接近最终解决方案的可交付件。它允许我们对最终用户和其他项目涉众演示应用程序，或者使这些人员可直接使用应用程序，从而提供关于评价的快速反馈。前进的方向正确吗？项目涉众对我们目前为止的工作满意吗？我们需要更改目前为止实施的功能吗？最后，还需要实施哪些附加功能来增加业务价值？通过回答这些问题，我们将有可能在项目涉

众之间建立信任的关系，而开发的系统将会更满足这些项目涉众的需要。因此就不会设计多余的思路或添加对最终用户无用的功能。

第二个需要是利用演示，并向调整计划提供反馈。不需要依靠评估规范（诸如需求规范、设计模型或计划），我们只需要评估至今为止开发的代码的工作情况。这说明我们必须使用测试结果并向项目涉众演示工作代码，来确定工作情况。这样可对进度、团队的进展速度，以及是否需要更正课程有一个更好的了解，从而成功完成项目。然后使用此信息来修改更新项目的总体计划，并为下一次迭代制定详细计划。

第三个需要是接受并管理变更。对于需求、设计、实施和测试来说，现在的应用程序过于复杂，无法在第一次就完全符合需求。相反，最有效的应用程序开发方法是接受更改的必然性。通过及时和持续的反馈，了解该如何改进应用程序，而迭代方法提供了以增量方式实施这些更改的机会。所有这些更改都要求准备好流程和工具才能进行管理，以便可以有效管理更改而不会阻碍创造力。

第四个需要是在生命周期早期排除主要风险，如图 2-10 所示。必须尽早处理主要的技术、业务和编程风险，而不是将风险的解决推迟到项目结束。可通过不断评估正面临的风险，并处理下一次迭代中的首要风险来完成此需要。在成功的项目中，早期迭代包括项目涉众接受远景和高级需求，其中包括体系结构设计、实施和测试以减轻技术风险。保留决定要使用哪些可重用资产或商业软件的所需信息也是很重要的。

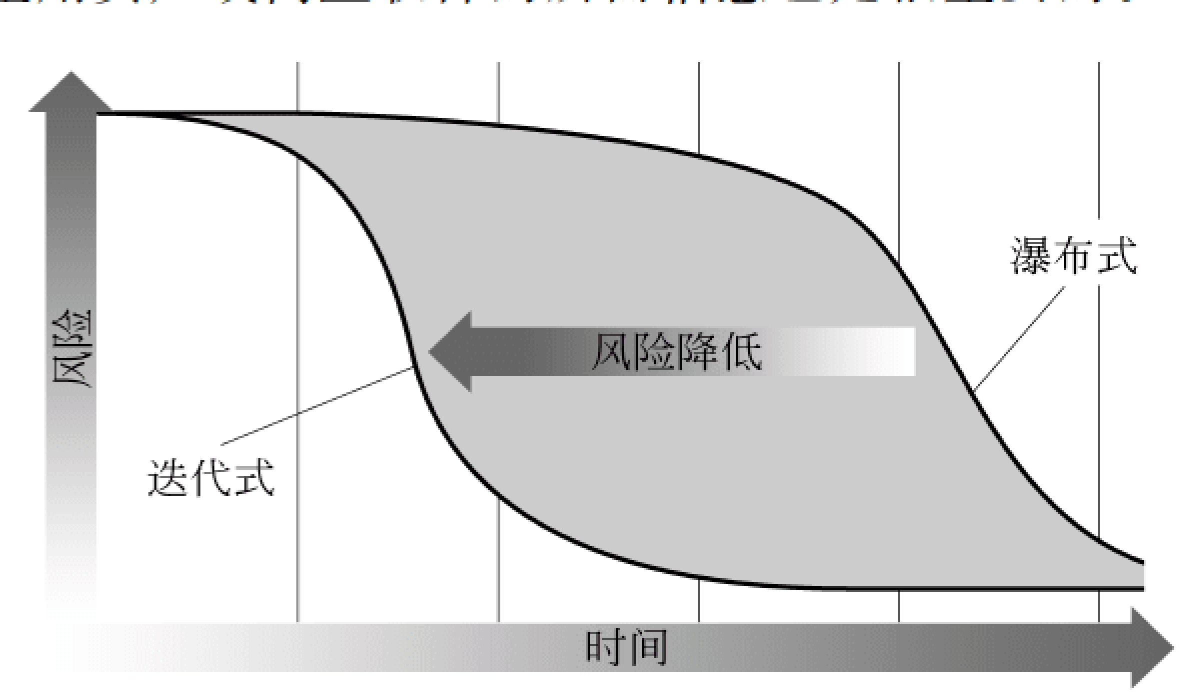


图 2-10　瀑布和迭代开发项目的风险消减一览图

迭代开发的主要目标是及早降低风险。可通过分析、划分优先级和处理每次迭代中的首要风险来完成此目标。

一个典型的反模式（即过去的导致项目失败的软件开发实践）是在整个生命周期开始前制定详细的计划，然后通过计划跟踪变化。另一个反模式是依赖于评审规格来在项目的前三分之二处评价项目状态，而不是评估测试结果和工作软件的演示情况。

2.7.5　提高抽象级别

好处：提高生产力，降低复杂度。

模式：复用已有资产，使用高级工具和语言来降低产生的文档数量，复原能力、质量、可理解性和复杂度控制的架构设计师。

反模式：直接将含糊的高级别需求变为定制的代码。

在软件开发中面临的一个主要问题是复杂性。我们知道降低复杂性对生产力有很大影响，在更高的抽象级别上工作将降低复杂性，并能促进沟通，其中一个降低复杂性的有效方法是重用现有资产，诸如可重用组件、旧系统、现有业务流程、模式或开放源代码软件。以下是在过去10年中对软件行业有重大影响的重要重用示例：

- 重用中间件，例如数据库、Web服务器和门户网站，以及最新的中间件。
- 开放源代码软件，提供许多可利用的小型和大型组件。

名词解释：中间件

中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源。中间件位于客户机/服务器的操作系统之上，管理计算资源和网络通信。是连接两个独立应用程序或独立系统的软件。相连接的系统，即使它们具有不同的接口，但通过中间件相互之间仍能交换信息。执行中间件的一个关键途径是信息传递。通过中间件，应用程序可以工作于多平台或OS环境，如图2-11所示。

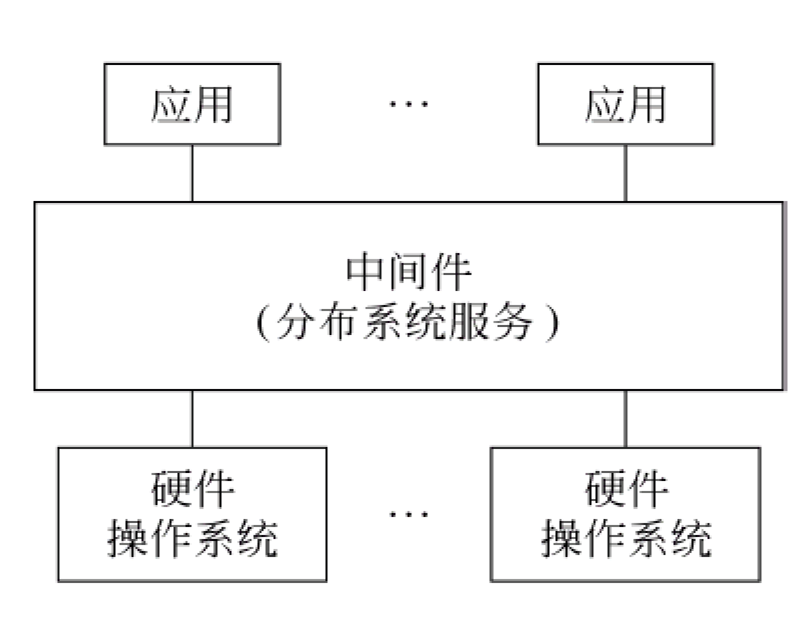


图2-11　中间件应用模型

中间件可分为数据访问中间件、远程过程调用中间件、消息中间件、交易中间件、对象中间件等。中间件应该具备两个关键特征：首先要为上层的应用层服务，这是一个基本条件；此外，又必须连接到操作系统的层面，并保持运行工作状态。具备了这样两个特征才能称为中间件。

最早具有中间件技术思想及功能的软件是IBM的CICS，但由于CICS不是分布式环境的产物，因此人们一般把Tuxedo作为第一个严格意义上的中间件产品。Tuxedo是1984年在当时属于AT&T的贝尔实验室开发完成的，但由于分布式处理当时并没有在商业应用上获得像今天一样的成功，Tuxedo在很长一段时期里只是实验室产品，后来被Novell收购，在经过Novell并不成功的商业推广之后，1995年被现在的BEA公司收购。尽管中间件的概念很早就已经产生，但中间件技术的广泛运用却是在最近10年之中。BEA公司1995年成立后收购Tuxedo才成为一个真正的中间件厂商，IBM的中间件MQ Series也是20世纪90年代的产品，其他许多中间件产品也都是最近几年才成熟起来。国内在中间件领域的起步阶段正是整个世界范围内中间件的初创时期。东方通科技早在1992年就开始中间件的研究与开发，1993年推出第一个产品TongLINK/Q。而中科院软件所、国防科技大学等研究机构也对中间件技术进行了同步研究。可以说，在中间件领域，国内的起步时间并不比国外晚许多。

更进一步来说，Web Service会对重用带来很大影响，在图2-12中，因为它提供了在不同平台并在消费者和服务供应商耦合松散的情况下重用功能的主要组块的简单方法，这意味着能更方便地利用服务的不同组合来处理业务需要，还可通过开放标准(如RAS、UDDI、SOAP、WSDL、XML和UML)来促进重用。

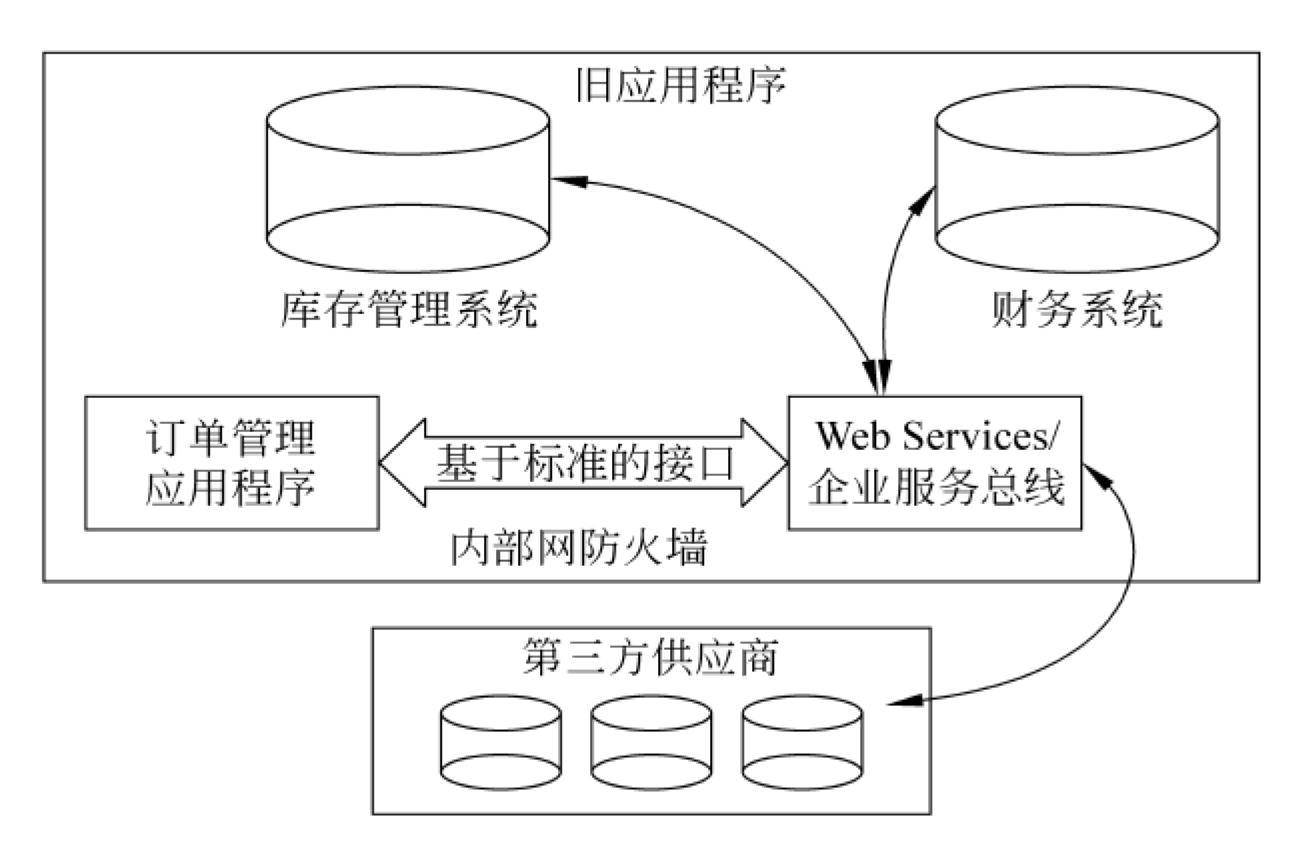


图 2-12　通过面向服务的体系结构来重用现有资产

名词解释：Web Service

Web Service 顾名思义就是一个运行在 Web 上的服务。这个服务通过网络为程序提供服务方法，类似一个远程的服务提供者。例如，一个提供天气预报的网站需要随时更新天气情况，在 Web 上挂上一个随时查询最新天气情况的服务。程序就可以从这个服务上获取到当前最新的天气信息。

Web Service 就是一个应用程序，它向外界暴露出一个能够通过 Web 进行调用的 API。这就是说，能够用编程的方法通过 Web 来调用这个应用程序。把调用这个 Web Service 的应用程序叫做客户。例如，想创建一个 Web Service ，它的作用是返回当前的天气情况。那么你可以建立一个 ASP 页面，它接受邮政编码作为查询字符串，然后返回一个由逗号隔开的字符串，包含了当前的气温和天气。要调用这个 ASP 页面，客户端需要发送下面的这个 HTTP GET 请求：

http://host.company.com/weather.asp? zipcode=20171

返回的数据就应该是这样：

21，晴

这个 ASP 页面就应该可以算作是 Web Service 了。因为它基于 HTTP GET 请求，暴露出了一个可以通过 Web 调用的 API。

Web Service 平台是一套标准，它定义了应用程序如何在 Web 上实现互操作性。读者可以用任何喜欢的语言，在任何喜欢的平台上写 Web Service ，只要可以通过 Web Service 标准对这些服务进行查询和访问。

重用的其中一个问题是在开发时，两个组件需要知道彼此的存在。面向服务的体系结构可通过提供称为松散耦合的服务来减轻此问题的影响：服务的消费者可动态找到服务的供应商。因此，可将现有组件或旧系统包含在服务中，允许其他组件或应用程序通过基于标准的接口，不必依赖平台和实施技术即可动态地访问那些功能。

降低复杂性并促进沟通的另一个方法包括利用高级工具、框架和语言：

① 标准语言（如 UML）和快速应用程序语言（如 EGL）提供表达高级别构造（如业务流程和服务组件）的功能，便于针对高级别构造进行协作，同时隐藏不必要的细节。

② 设计和构造工具，可自动从高级别构造过渡到工作代码：

- 提供向导，通过生成代码并启用代码片段来自动设计、构造和测试任务。
- 通过集成开发、构建和测试环境，将集成和测试转换为无缝开发任务。

③ 组合包管理工具，将多个项目的财务和其他方面管理作为一个实体，而不是一组单独的实体。

高级别工具可以图形化方式捕获重要的建模信息，图形化是概括和表达此信息的最有效的方法，而且更能吸引用户的注意力。

管理复杂性的第三个方法为注重体系结构，定义业务或者开发系统或应用程序。在软件开发中，旨在项目早期设计、实施和测试体系结构。在项目早期要注重以下目标：

① 定义高级别构建模块和最重要的组件及其功能和接口。

② 设计常见问题的解决方案，例如如何处理持久性或垃圾回收问题。

通过尽早确定体系结构，可提供系统的框架结构，使其在更多人员、组件、功能和代码添加到项目时更为简单。还可确定利用哪些可重用资产，以及需要定制构建哪些系统。

这一原则的反模式是直接从模糊的需求开始进行定制代码和设计。由于没有进行抽象，在代码级而不是一个高级级别上进行了很多讨论，这就失去了很多复用的机会。掌握的非形式化的需求和其他信息需要很多决定和规范来不断重新访问，而对架构的忽略或不重视造成了在项目后期的返工。

2.7.6 持续关注质量

好处：更高的质量，更早地了解进度和质量。

模式：团队对最终产品负责。尽早地进行测试，并与可演示功能的集成同步进行不断的测试。逐渐地建立测试自动化。

反模式：推迟集成测试直到所有代码都完成并且单元测试也已完成。详细检查所有工件以发现问题，但不进行部分实现和测试。

提高质量不是简单的满足需求，或是生产出满足用户需要和期望的产品。质量还包括确定用于证明实现质量的度量和标准，以及实施一个流程以确保产品已达到所期望的质量水平并可重复使用和管理。

保证高质量不仅需要测试团队的参与；还需要整个团队关注质量。这涉及了所有团队成员以及生命周期的所有部分。分析师对确定需求的可测试性负责，而且还要明确对测试的需求。开发人员设计应用程序时需要具有测试意识，并且必须负责测试自己的代码。

管理员需要确保合适的测试计划到位，并确保构建测件及执行所需测试的资源到位。测试员是质量专家。他们对团队其他成员就理解软件质量问题进行指导，而且他们负责所有的产品测试（包括功能、系统及性能等）。遇到质量问题时，每个团队成员都应参与进来帮助解决问题。

迭代开发的主要优势就是尽早地不断地测试成为可能，如图 2-13 所示。项目结束之前，因为最重要的功能很早就已经实现，软件的最初版本可能已经启动且运行了数月，在此同时应该对其进行了数月的测试。大部分采用迭代式开发的项目表明改善过程带来了切实的质量提高。

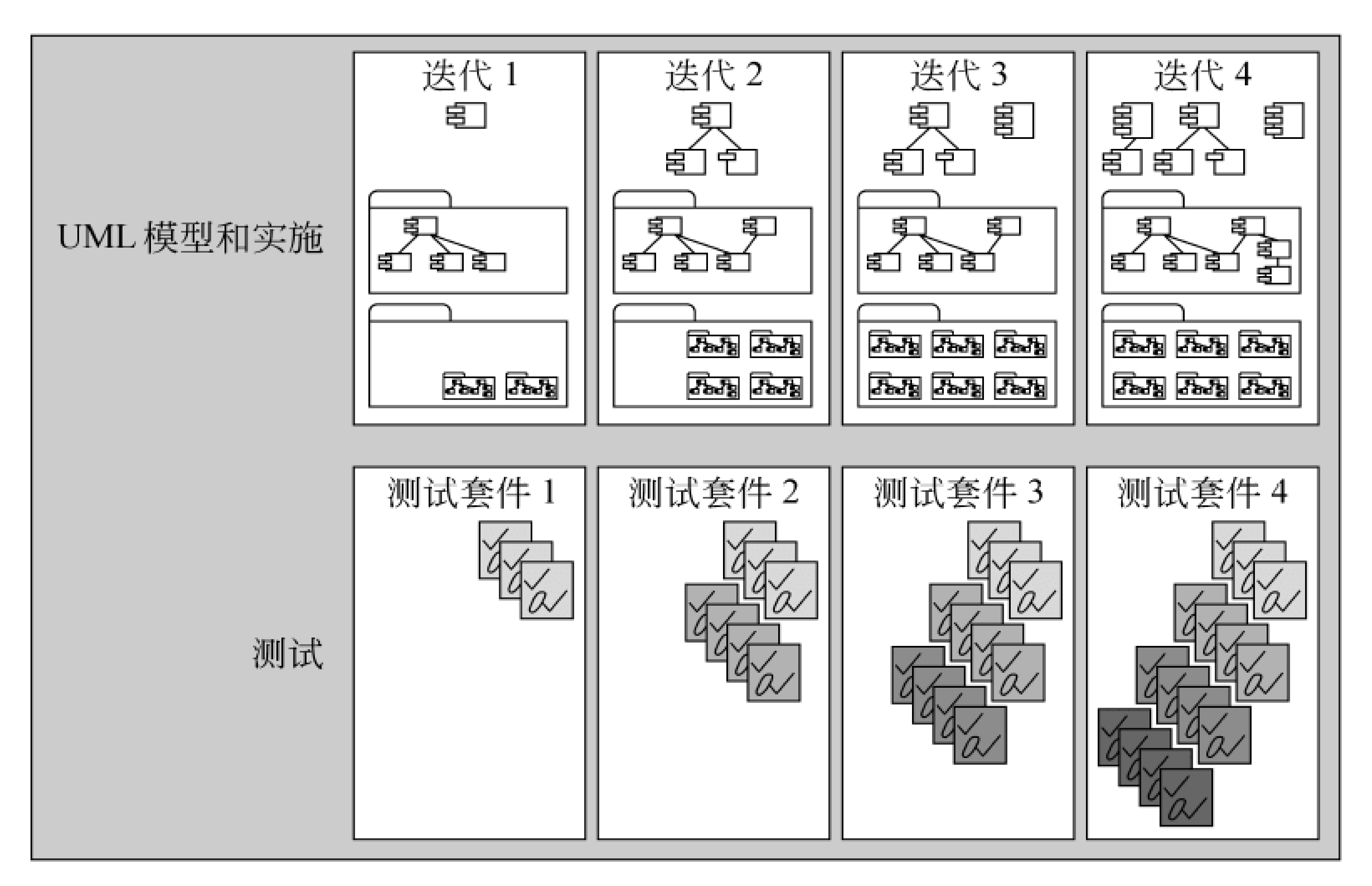


图 2-13　测试尽早启动并在每一次迭代中扩展

增量构建应用程序时，我们还应该递增构建测试自动化来及早发现缺陷，同时使前期投资最小化。在设计系统时，还需要考虑到如何对其进行测试，这样可以帮助提高自动化测试能力。还可以直接从设计模型生成测试代码，这样可以节省时间，有助于开展早期测试，减少软件缺陷提高测试质量。自动化测试已经成为很多项目团队关注的主要领域，自动化测试的目标是使所有代码的测试自动化，并且测试在编写代码之前就已编写好（即测试优先设计）。

迭代开发使早期测试成为可能。每次迭代中开发的软件在构建时都要进行测试。回归测试保证了新的迭代添加功能后不会引入新的缺陷。

这一原则的反模式是对所有中间工件进行深入详细的回顾分析，这不利于提高生产效率，因为它延迟了应用测试，也就延迟了主要问题的识别。另一个反模式是在进行整体测试前完成所有单元测试，同样也延迟了主要问题的识别。

2.8　RUP4+1 视图

1995 年，Philippe Kruchten 在《IEEE Software》上发表了题为《The 4+1 View Model of Architecture》的论文，引起了业界的极大关注，并最终被 RUP 采纳。如图 2-14 所示 RUP 所建议的五个视图中，场景视图是以使用场景的观点来驱动其他四个视图，所以又称为 4+1 架构视图的模型。

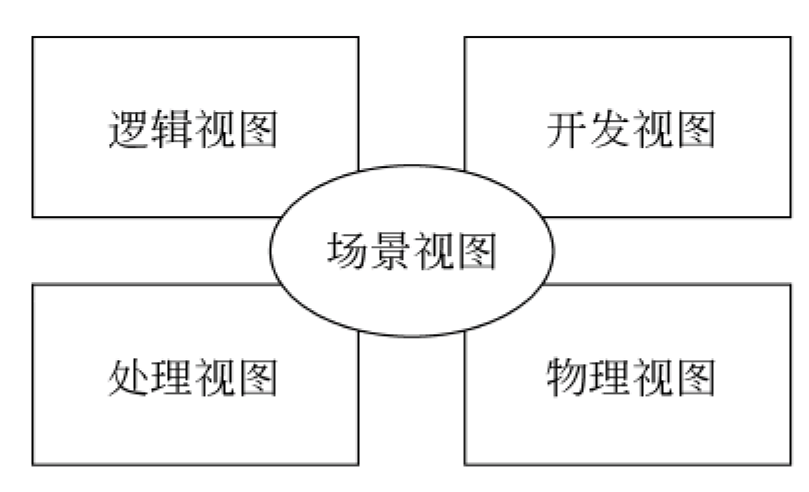


图 2-14　4+1 架构视图

RUP 的 4+1 视图方法为实现成功的软件架构给出了更加切实可行的方法。

RUP 的开发模式是以架构为中心(Architecture-Centric)的流程,当组织或团队对架构的呈现方式已经取得共识,下一个议题就是学会架构设计的流程与规划(architectural design process)。

RUP 利用从五个架构视图的方式来简化对架构的描述(或称之为抽象化)。每一个视图会涵盖与之相关的考虑点,而忽略跟该视图无关的东西。对每一个视图,需要清楚找出:

- 希望专注的考虑点和参与者。
- 视图中所捕捉并呈现的元素和它们之间的关系。
- 建立视图元素与其他视图元素之间的关系。
- 建立视图的最佳流程。

该方法的不同架构视图承载不同的架构设计决策,支持不同的目标和用途。

① 逻辑视图:当采用面向对象的设计方法时,逻辑视图即对象模型。它是设计模型(design model)的抽象概念,用来找出在分析与设计阶段时的类别(class)、子系统(sub system)、主要套件(package)。例如订购、订购细节、顾客、产品和产品型号等对象。

② 开发视图:描述软件在开发环境下的静态软件的模块(module)、源代码(source code)、数据文件、组件(component)、可执行(executable)码和其他工作成果等组织。它专注再次开发、软件资产管理、可重用性(reuse)、外包和添加组件等议题上。例如订购子系统的原始程序代码。

③ 处理视图:描述系统的并发和同步方面的设计。处理试图会视项目规模的大小,而有不同的应对与处理方式。例如,订购系统如何处理同时上线客户的人数超过 5000 人,仍能保持系统的稳定度与效能(如 3 秒内完成订购的交易)的处理。

④ 物理视图:描述软件如何映射到硬件,反映系统在分布方面的设计。例如,在 Windows 2000 上,安装了 IIS 以及 .NET Framework、MTS(Microsoft Transaction Service)等应用服务器,然后在其内配置了订购系统,并连接另一台同为 Windows 2000 系统、安装了 SQL Server 2000 的数据库系统。

2.9 RUP 裁剪

RUP 是一个通用的过程模板,包含了很多开发指南、制品、开发过程所涉及的角色说明,由于它非常庞大,所以对具体的开发机构和项目,用 RUP 时还要做裁剪,也就是要对 RUP 进行配置。RUP 就像一个元过程,通过对 RUP 进行裁剪可以得到很多不同的开发过程,这些软件开发过程可以看作 RUP 的具体实例。RUP 裁剪可以分为以下几步:

① 确定本项目需要哪些规程。RUP 的九个核心规程并不是都需要,可以取舍。

② 确定每个规程需要哪些制品。

③ 确定四个阶段之间如何演进。确定阶段间演进要以风险控制为原则,决定每个阶段要那些规程,每个规程执行到什么程度,制品有哪些,每个制品完成到什么程度。

④ 确定每个阶段内的迭代计划。规划 RUP 的四个阶段中每次迭代开发的内容。

⑤ 规划规程内部结构。规程涉及角色、活动及制品，复杂程度与项目规模即角色多少有关。最后规划规程的内部结构，通常用活动图的形式给出。

RUP 裁剪案例

对于像中兴这样的大公司，为了利用一切可用条件最大限度扩展生产规模，有很多产品的生产经常需要外包，但他们的产品又经常需要针对客户的个性化需求进行定制。如何既能满足客户需求，又能保证产品质量，这就需要在外包管理中，跟踪、监控开发过程，使用统一的沟通语言、技巧和方法，还要有科学的管理水平。

他们认为 RUP 剪裁就是很好的外包管理解决方案。那么中兴是如何通过对 RUP 的定制，使它适合软件外包管理的要求呢？总体而言，软件的外包管理包括承包方的选择和外包开发的管理两大部分，由于篇幅所限，本文仅讨论外包开发管理。

软件项目外包开发有其显著特点，那就是承包方自己承担软件开发和人员的管理，但软件产品和与双方都相关的软件过程则由委托方和承包方共同管理。因此，外包项目的管理可以分为 3 个阶段：项目启动、项目开发、项目验收，其中项目启动对应 RUP 的初始阶段，项目开发对应细化和构造阶段，项目验收则对应交付阶段。

在 RUP 剪裁方面，用于管理自身开发和用于管理承包方开发有很大的不同。例如一方面，开发工作由承包方来做，所以在外包开发阶段，大多数核心工程规程为空，但另一方面，承包方的开发过程不透明又非常危险，难以监控开发进度，所以多设立检查点非常必要。

另外需要注意的是外包的软件项目，往往是中兴自己的重大项目的子项目。这时，中兴还会考察双方软件过程的“兼容性”，以避免在最后才进行集成，而是在早期的里程碑就应持续不断地进行集成并进行相关的测试和反馈。

2.10 实践经验

在 2.9 节讨论了软件过程开发的相关理论。理论是用来指导实践的，RUP 作为一种著名的软件过程，实施照样离不开理论的指导。深刻地认识 RUP 必须不懈地学习和实践。但是，倘若缺乏理论指导，RUP 实施将难免有些“莽撞”，如同瞎子摸象、囫囵吞枣，这恐怕也是有些人使用 RUP 以及其他软件工程方法收获不大的根本原因所在。

这里以上述理论为指导谈一些 RUP 实施中的具体问题。首先从整个 RUP 实施的角度，讨论其宏观过程，并重点分析过程开发理论的指导作用。

在开发组织中引入 RUP 及其支持工具，原因是因为它们可以提高项目质量，从而得到实际的商业利益。但是 RUP 的引入对开发组织来说，意味着改变（或部分改变）以前的工作方法，其中必然会有一定风险。

软件过程也是软件在进行需求采集、分析、设计之后开发出适合团队实际情况的 RUP 实施方案。实施方案评审通过之后，就成为指导团队协作的管理依据，这时 RUP 实施已完全明朗。

下面对上述过程进行工程化的描述。在项目开始时引入 RUP，共需如下 5 个步骤：

① 评价(assess)。评价开发团队以前在软件过程方面遇到的问题，明确过程实施的目标。

② 计划(alas)。计划 RUP 的具体实施过程，包括相关培训。

③ 配置和定制(configure and customize)，剪裁和定制 RUP 使它符合项目的实际情况。

④ 执行(execute)。进行培训并启动项目，在项目中运用定制的 RUP。

⑤ 评估(evaluate)。评估 RUP 的实施情况为以后改进做准备。

由上面的步骤可以看出，RUP 的实施不是一个单纯的过程开发与执行，而是过程的再工程与执行。再工程(reengineering)是一个工程过程，它通过逆向工程、重构和正向工程的组合，将现有系统重新实现为一种新的形式。RUP 包含了众多的软件开发最佳实践，以及详尽的工作指南，在 RUP 的基础上进行再工程是明智的。由此还可以看出软件过程再工程理论，对指导具体 RUP 实施工作是非常有意义的。

阅读材料

再工程主要出于如下愿望：在市场上要提高产品的竞争力，在技术上要提高产品的质量。但这两种愿望无法靠软件的维护来实现，一是软件的可维护性可能极差，实在不值得去做；二是即使软件的可维护性比较好，但也只是治标不治本。再生工程就是对已有软件进行全部或部分的改造，赋予软件新的活力。

例如在对待一个不良之徒时，可以进行思想教育并给予他关心和帮助，这种方式类似于“软件维护”；也可以把他关进监狱，送去劳改，这种方式相当于软件的“再生工程”；如果此人罪大恶极，就毙掉算了。

再工程与维护的共同之处是没有抛弃原有的软件。如果把维护比作“修修补补”，那么再工程就算是“痛改前非”。再工程并不见得一定比维护的代价要高，但再工程在将来获取的利益却要比通过维护得到的多。

再工程主要有三种类型：重构、逆向工程和前向工程。

逆向工程来源于硬件世界。硬件厂商总想弄到竞争对手产品的设计和制造“奥秘”。但是又得不到现成的档案，只好拆卸对手的产品并进行分析，企图从中获取有价值的东西。软件的逆向工程在道理上与硬件的(逆向工程)相似。但在很多时候，软件的逆向工程并不是针对竞争对手的，而是针对自己公司多年前的产品。期望从老产品中提取系统设计、需求说明等有价值的信息。

前向工程也称预防性维护，由 Miller 倡导。他把这个术语解释成“为了明天的需要，把今天的方法应用到昨天的系统上”。乍看起来，主动去改造一个目前运行得正常的软件系统，简直就是“惹是生非”。但是软件技术发展如此迅速，与其等待一个有价值的产品逐渐老死，还不如主动去更新，以获取更大的收益。其道理就同打预防针一样。所以，预防性维护是“吃小亏占大便宜”。

需要说明的是实施 RUP 最好在项目开始时，而不是在项目进行一半或结束时再实

施，这样可以避免在项目过程中，工作方法的改变引起的复杂问题。

RUP 适用于规模比较大的软件项目和大型的软件开发组织或团队，提供了在软件开发中涉及的几乎所有方面的内容。但是对于中、小规模的软件项目，开发团队的规模不是很大，软件的开发周期也比较短。在这种情况下，完全照搬 RUP 并不完全适用。

可以通过两种方法来改进 RUP，使得它适合中、小型软件开发项目。一种方法是，对 RUP 进行适当裁剪，例如结合开发技术的特点从项目管理、架构设计、开发和测试等方面对 RUP 进行裁剪，应用到小型项目团队开发应用系统的过程中。在项目管理中，正确定义团队角色，采用迭代式的开发方式和重视风险管理；在架构设计中，针对开发技术的特点，指出了从三个不同方面来设计软件架构；在开发和测试中，正确对待各个阶段的集成和测试。另一种方法就是将 XP 与 RUP 相结合来开发小型项目。

以下是 RUP 原理应用的两个项目例子：

① 如果要写一本《基于 RUP 的软件测试实践》的书，首先的目标是可交付工件（本书）。这本书的内容和结构、关键章节、教学方法都是这本书中未知或不可预期的例子。首先要确定目标读者，确定本书的基本构架，并且为每一章写一个简短的描述。之后，再细化每一章节，突出关键内容，在撰写其他章节之前，检查已写好基本就绪的章节，即进行单元测试。再一步是根据计划的构架完成全书，对全部书稿进行检测核对、修改，即系统测试、功能测试。最后再交送给出版社。

② 进行行业调查和研究。首先需要了解不利于开展调查的问题，制定一个粗略的时间安排和调查期间的资源需求。之后随着对问题认识的深化，尝试用不同的方法来减小研究的第一不确定性。消除了第一高风险后，再继续进行研究。最后撰写研究报告，获得客户的同意。它的研究方法都基于同 RUP 类似的原理。

2.11 小　　结

软件开发的方法论已经成为现代软件工程开发过程中不可缺少的一个重要部分。目前，RUP 是被广泛采用的一种方法。与传统的瀑布模型开发方法相比，RUP 有降低风险、适应需求变化等优点，并且为软件开发的整个生命周期提供了基础框架和指导。

本章介绍了 RUP 的概念，从三个方面阐述了 RUP 的理念，并进一步介绍了 RUP 的发展史、六个最佳软件实践经验、RUP 的三个重要特点。指出 RUP 具有迭代式的增量开发、由用例驱动、以架构设计为核心的三个鲜明特点，这使得 RUP 非常适宜于开发复杂、技术难度大、需求多变、高风险的项目。作为 RUP 的六个最佳软件开发实践的主要更新，还阐述了一组新的原则，可用来刻画构建、部署和演化软件密集系统的成熟方法。最后还介绍了 RUP4＋1 视图和 RUP 裁剪。RUP 又是可裁剪的软件开发过程框架，各组织可以根据自身及项目特点对 RUP 进行裁减，在某些情况下 RUP 甚至可以蜕化为瀑布式开发模型。

RUP 具有很多长处：提高了团队生产力，在迭代的开发过程、需求管理、基于组件的体系结构、可视化软件建模、验证软件质量及控制软件变更等方面，针对所有关键的开发

活动为每个开发成员提供了必要的准则、模板和工具指导，并确保全体成员共享相同的知识基础。它建立了简洁和清晰的过程结构，为开发过程提供较大的通用性。但同时它也存在一些不足：RUP 只是一个开发过程，并没有涵盖软件过程的全部内容，例如它缺少关于软件运行和支持等方面的内容；此外，它没有支持多项目的开发结构，这在一定程度上降低了在开发组织内大范围实现重用的可能性。可以说 RUP 是一个非常好的开端，但并不完美，在实际的应用中可以根据需要，对其进行改进并可以用面向对象的软件过程等其他软件过程的相关内容对 RUP 进行补充和完善。

习题与思考

1. 什么是 RUP？RUP 可以从哪三个方面理解？
2. 简述 RUP 的发展史。
3. 简述 RUP 的特点。
4. 简述 RUP 包括的软件开发中的六大经验。
5. RUP 的核心概念是什么？请说明每个核心概念的基本内容。
6. RUP 把软件开发生命周期划分为多个循环，每个循环包含哪些内容？
7. 什么是构架设计？为什么 RUP 要强调软件开发以构架设计为中心？
8. 简述 RUP 的 4+1 视图。
9. 什么是 RUP 裁减？简述 RUP 裁减的步骤。
10. 为什么 RUP 是由用例驱动的？

第  章 RUP 测试概论

问题：有关汽车生产过程质量

让我们来设想一个场景：在一家汽车销售公司里，汽车销售员正在为准客户的您推销两款汽车，其中一款是由某公司引入当时世界上最先进的生产线和工艺流程生产的产品，而另一款是由厂家技术精湛的师傅花了一个多月的时间用车床加大锤手工精制而成。排除其他购买因素，在汽车的质量上，您认为哪款最好呢？

我们可以作个简单的分析，第一辆车的质量是由汽车生产线和生产工艺本身决定的，每一辆同型号车的质量应该完全相同，因为它是由汽车生产的过程质量决定的。对第一辆车的质量，通过了解市场上同型号车的质量状况，便可以做到心中有数。

而第二辆车的质量在很大程度上则依赖于生产汽车的师傅水平，不同的师傅生产出的汽车质量可能相差很大。所以需要花一番工夫弄清楚师傅的资质背景，从而判断汽车的质量。由此可见，对第一辆车的信任，来自于过程质量。而汽车作为日常消费商品被大规模生产和销售的基础也是过程质量，它使汽车生产的规模经济成为可能。

一个软件产品也需要过程质量，一般最直接的就是通过测试流程、测试规范来保证质量。当然还有很多的环节还会发生错误，例如配置管理环节、版本的管理环节等，这些也需要相关的支持来保证软件的质量。所以说软件质量保证不应该只是在某一个环节上，而应该是整个的流程，应该全面地去改进流程来保证质量。这就是本章要讲的 RUP 测试理念的核心。

传统的软件测试已经突显出越来越多的问题，软件工程和测试技术的快速发展也导致许多理论技术不再适用于现代软件开发。而 RUP 的整体结构也一直在不断地进行修改，它与时代技术得到了同步发展。其中 RUP 的测试方法通过不断改进，逐渐形成了现在的测试体系。按照 RUP 的理论，测试是系统开发中的一个重要部分。RUP 第五个最佳实践经验，即持续地质量保证主要说明了所有的开发活动和工件都需要通过持续的测试和复审来检查质量，测试决不仅仅是软件构建之后的一个阶段。

在 2002 年以前，RUP 的重点是在传统的计划、规格说明和测试的执行上，并且将大部分注意力放在了测试自动化上。在 2003 年中，测试流程发生了相当大的变化。它转向了基于探索性测试的方法，此方法中的认识系统、设计和执行测试并行活动。在系统文档经常改变的情况下，不将重点放在设计和编写基于文档的测试用例这些耗时的任务上，而且系统文档不应当被唯一地用作测试依据。

学习本章要求读者重点理解和掌握 RUP 测试的成功经验和测试理念，并了解与质量保证的关系。同时掌握 RUP 测试流程和评测方法，这将是下一步学习 RUP 测试实践的基础。

3.1 软件测试

针对传统软件测试容易导致项目进度难于控制、风险控制能力较弱和由于测试较晚而导致开发费用增加的问题，RUP 的软件测试提出了三大成功经验：尽早测试、连续测试、自动化测试。

3.1.1 传统软件测试的问题

传统的软件测试流程一般是先在软件开发过程中进行少量的单元测试，然后在整个软件开发结束阶段，集中进行大量的测试，包括功能和性能的集成测试和系统测试。随着开发的软件项目越来越复杂，传统的软件测试流程不可避免地给工作带来以下问题：

1. 项目进度难于控制，项目管理难度加大

如图 3-1 所示，大量的软件错误往往只有到了项目后期，即系统测试时才能够被发现。解决问题所花的时间很难预料，经常导致项目进度无法控制，同时在整个软件开发过程中，项目管理人员缺乏对软件质量状况的了解和控制，加大了项目管理难度。

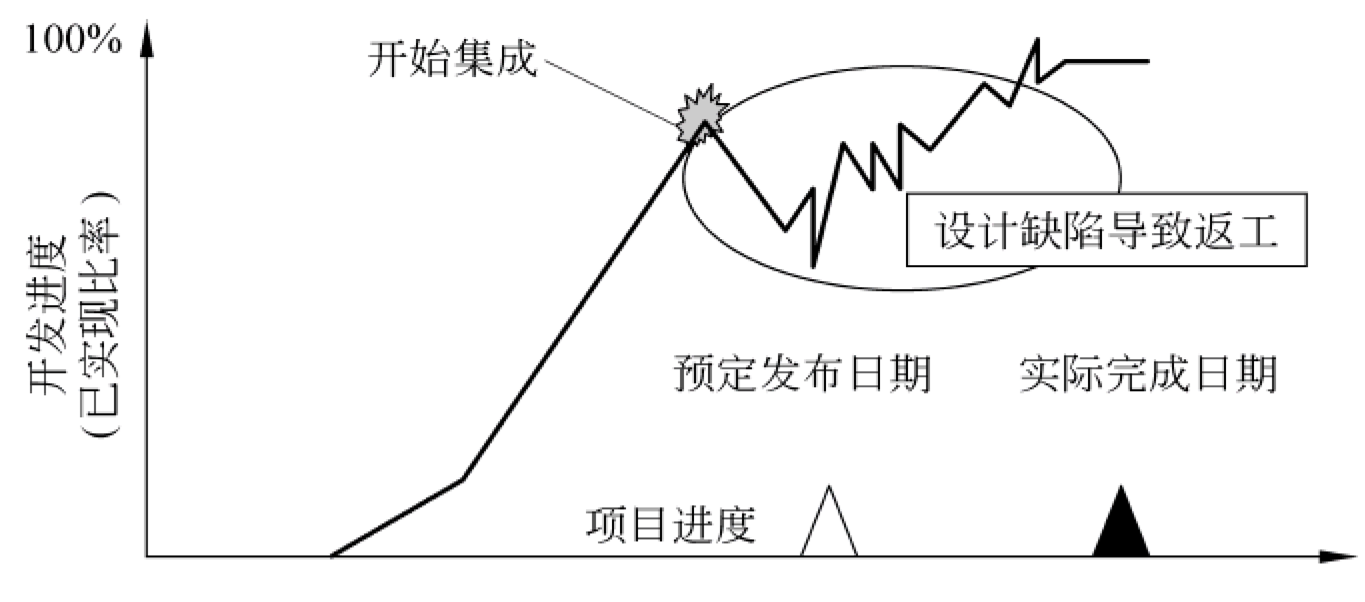


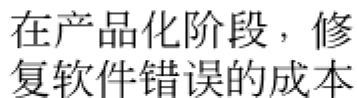
图 3-1 传统测试流程中存在的问题

2. 对于项目风险的控制能力较弱

项目风险在项目开发较晚的时候才能够真正降低。往往是经过系统测试之后，才能确定该设计是否能够满足系统功能、性能和可靠性方面的需求。

3. 软件项目开发费用超出预算

在整个软件开发周期中，错误发现得越晚，单位错误修复成本越高，如图 3-2 所示，错误延迟解决，必然导致整个项目成本的急剧增加。



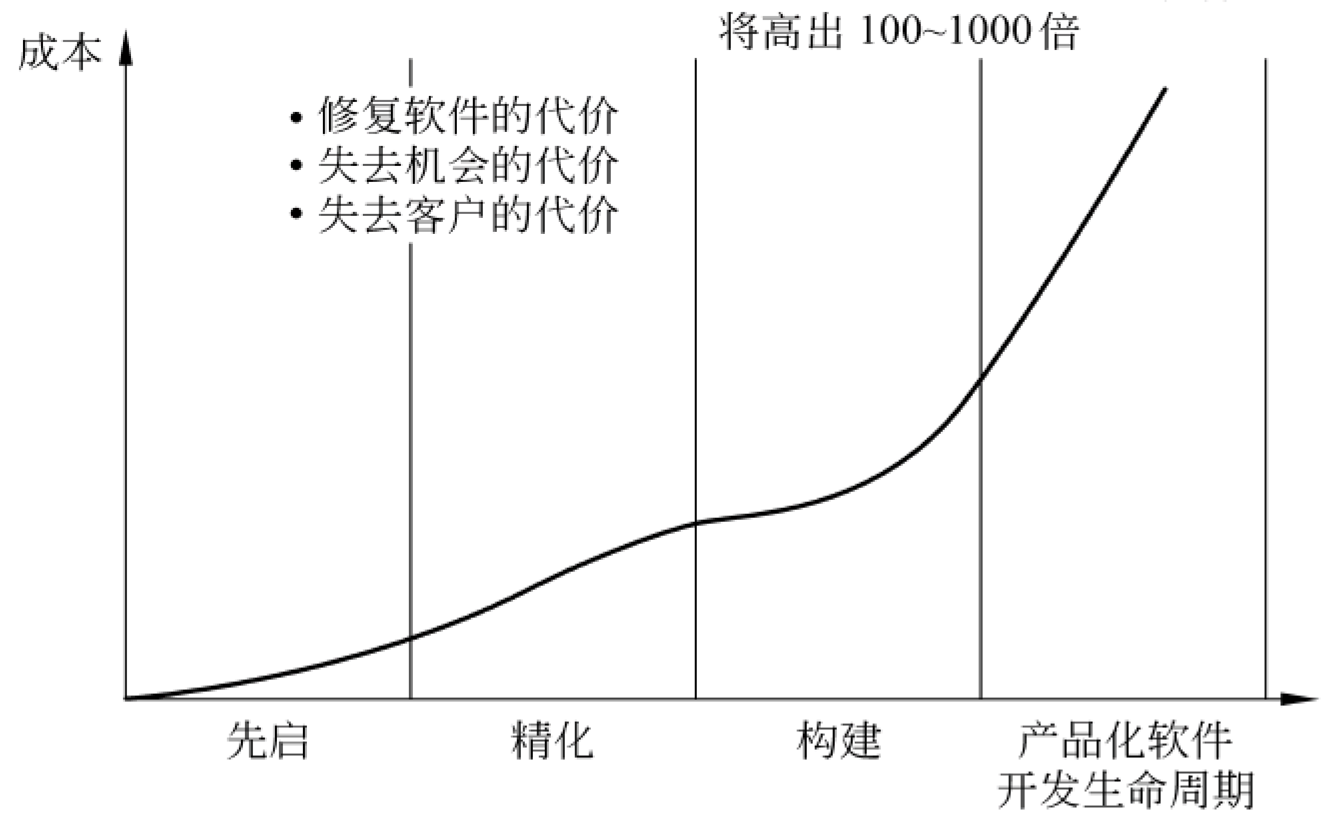


图 3-2　传统测试流程中存在的问题

3.1.2　基于 RUP 的软件测试成功经验

基于 RUP 的软件测试技术核心的三个成功经验是：尽早测试、连续测试、自动化测试。并在此基础上提供了完整的软件测试流程和一整套的软件自动化测试工具，最终能够做到：一个测试团队，基于一套完整的软件测试流程，使用一套完整的自动化软件测试工具，完成全方位的软件质量验证。

1. 尽早测试

尽早测试是指在整个软件开发生命周期中，通过各种软件工程技术，尽早地完成各种软件测试任务的一种思想。RUP 主要在以下三个方面提供了尽早测试的软件工程技术。

首先，软件的整个测试生命周期是与软件的开发生命周期基本平齐的过程，如图 3-3 所示，即当需求分析基本明确后，就应该基于需求分析的结果和整个项目计划来进行软件的测试计划；伴随着分析设计过程，同时应该完成测试用例的设计；当软件的第一个版本发布出来后，测试人员要马上基于它进行测试脚本的实现，并基于测试计划中的测试目的执行测试用例，对测试结果给出评估报告。这样，便可以通过各种测试指标实时监控项目质量状况，提高对整个项目的控制和管理能力。

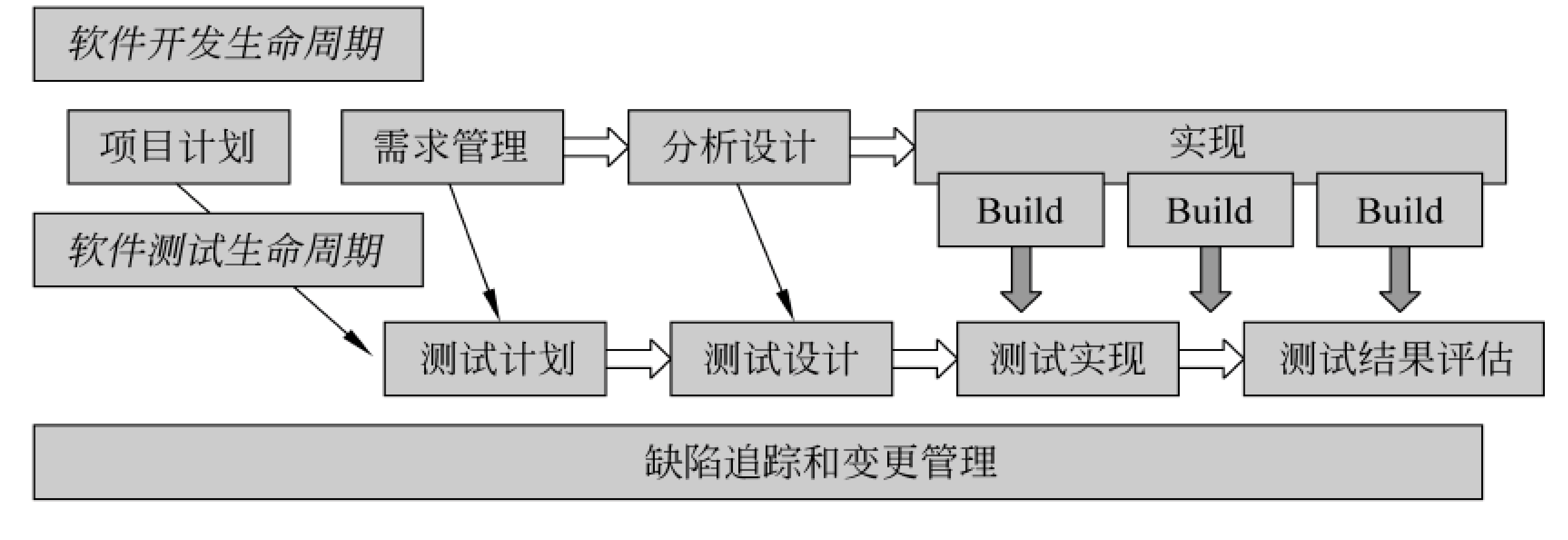


图 3-3　软件测试生命周期

其次，测试最重要的问题就是成本问题。传统开发阶段包括需求分析阶段、设计阶段、编码阶段、测试和最终的交付阶段。产品上线的过程中，如果发现问题越早，付出的代价越低。如果到了生产线以后发现问题，成本几乎是以前的 95 倍。因为这个时候会有很多流程，如果把问题带到了产品生产线环境里，所带来的费用是非常巨大的。国外一个研究机构发现这样一个规律：每个程序员每小时产生 4.2 个缺陷，这个数据根据不同的开发语言，不同的技术层面有所差异，这是一个大概平均值，而且大部分的缺陷产生于生产、开发阶段。在编码阶段产生的缺陷，到后面解决成本投入会越来越高。如果尽早发现缺陷，尽早解决它，就可以降低成本。

通过迭代式软件开发把原来的整个软件开发生命周期分成多个迭代周期，在每个迭代周期都进行测试，这样在很大程度上提前了软件系统测试发生的时间，可以在很大程度上降低项目风险和项目开发成本。

最后，RUP 的尽早测试成功经验，还体现在它扩展了传统软件测试阶段从单元测试、集成测试到系统测试、验收测试的划分，将整个软件的测试按阶段划分成开发人员测试和系统测试两个阶段，如图 3-4 所示，它把软件的测试扩展到整个开发人员的工作过程。通过提前测试发生的时间来尽早地提高软件质量、降低软件测试成本。

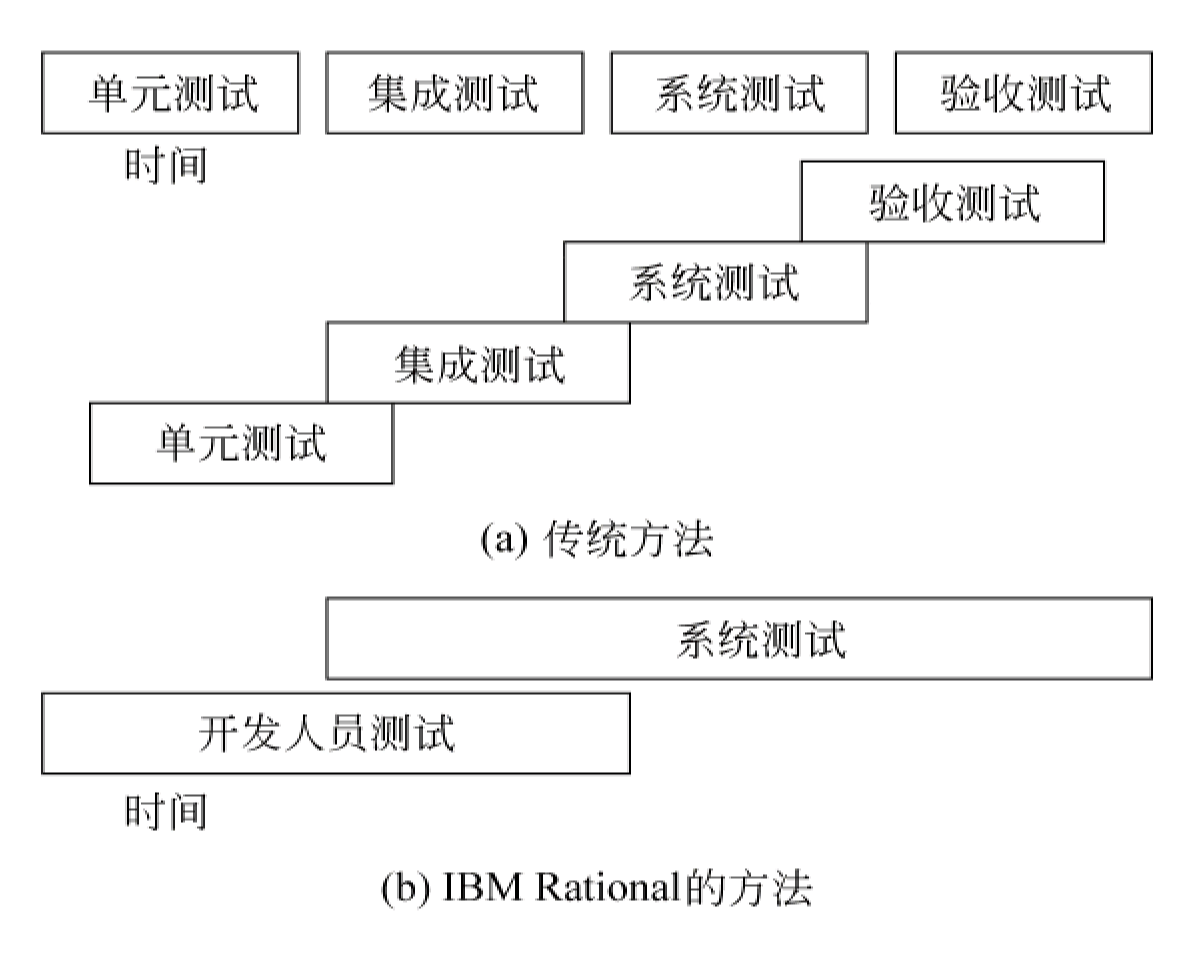


图 3-4　测试阶段的划分

2. 连续测试

连续测试是从迭代式软件开发模式中得来的。在迭代化的方法中，将整个项目的开发目标划分成为一些更易于完成和达到的阶段性小目标，这些小目标都有一个定义明确的阶段性评估标准。在每个迭代开始前，都要根据项目当前的状态和所要达到的阶段性目标制定迭代计划，而且每个迭代中都包括需求、设计、编码、集成、测试等一系列的开发活动，都会增量式集成一些新的系统功能。通过每次迭代，都会产生一个可运行的系统。通过对于这个可运行系统的测试，来评估该次迭代有没有达到预定的迭代目标，并以此为依据来制定下一次迭代的目标。由此可见，在迭代式软件开发的每个迭代周期，都会进行

软件测试活动。整个软件测试的完成，是通过每个迭代周期不断增量测试和回归测试实现的。

如图 3-5 所示，采用连续测试的软件成功测试经验，不但能够持续的提高软件质量、监控质量状态，同时也使系统测试尽早实现成为可能。从而有效地控制开发风险、降低测试成本，切实保证项目进度。

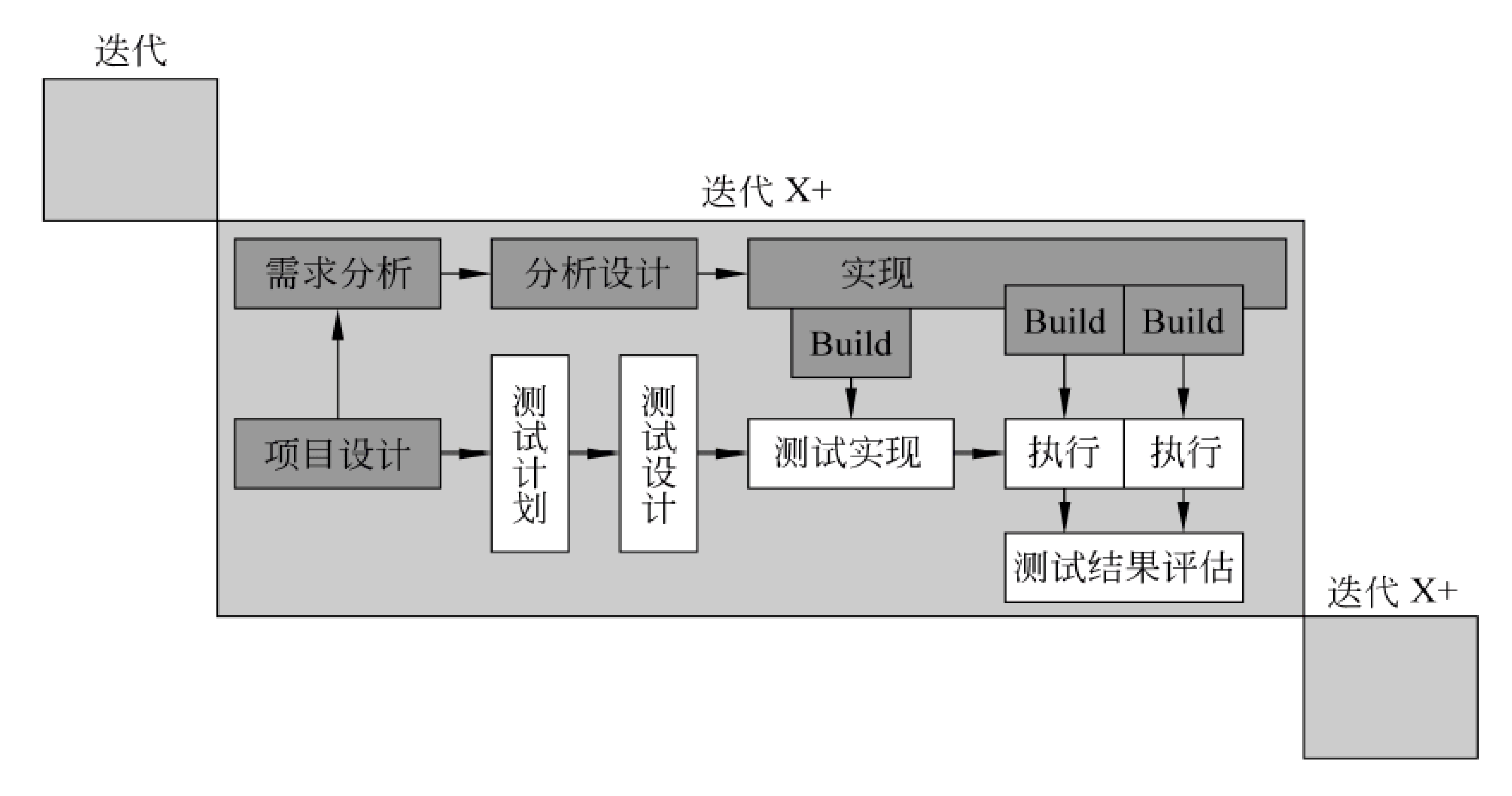


图 3-5　成功测试经验之二：连续测试

3. 自动化测试

在整个软件的测试过程中要想实现尽早测试、连续测试，可以说完善的测试流程是前提，自动化测试工具是保证。RUP 的自动化测试成功经验主要是指利用软件测试工具提供完整的软件测试流程的支持和各种测试的自动化实现。

为了使各种软件测试团队更好地进行测试，各大测试工具厂商在借鉴 RUP 的测试成功经验之外，还提供了一整套软件测试流程和自动化测试工具，使软件测试团队能够有条不紊地完成整个测试任务。

问题：自动化测试平台搭建和软件工具配置跟测试有什么关系？

分析：其实测试跟开发是类似的，有很多要递交的东西，例如测试计划、测试用例、测试数据、测试报告、测试中间件产生的状态，这些东西都是需要有一个测试的平台来管理。因为现代软件开发不是个人作战，而是一个团队来完成。很多时候，作为测试员，测试的不是一个人，而是一个团队或一个中心，很多人在里面需要一系列的产出物，这是需要有人管理的。而管理手段不是在大脑里或手中的笔记里，它没有办法沟通，一个一个信息孤岛也没有办法交流，更没有办法互相监测。很多做测试管理、测试开发和测试培训的人员，他们面对最大的挑战就是不知道现在进度在哪儿，也不知道挑战什么样子，也不知道手下什么样，这就是缺乏平台导致的。所以要有个平台负责从头到尾进行管理，这是自动化测试平台搭建和软件工具配置与测试的关系。

3.2 RUP 软件测试流程

RUP 软件测试流程不仅包含完整的软件测试流程框架，同时还提供了内嵌软件测试流程的测试管理工具的支持，包括完整的测试评测方法。

3.2.1 软件测试流程框架

不管做什么事情，都有一个循序渐进的过程，即从计划到策略再到实现。软件流程就是按照这种思维来定义的开发过程。它根据不同的产品特点和以往的成功经验，定义了从需求到最终产品交付的一整套流程。流程告诉我们该怎么一步一步去实现产品，可能会有那些风险，如何去避免风险等。由于流程来源于成功的经验，因此，按照流程进行开发可以使得我们少走弯路，并有效地提高产品质量，提高用户的满意度。

RUP 对测试管理流程进行了完整的定义，通常有一个角色和一个行动，这样会有一个固化的流程来帮助测试团队执行这个工作。RUP 通过这些来强调测试的进度和质量，通常会事先规定需求覆盖率和测试用例实现率，然后开发测试用例，看是否达到了覆盖率和实现率。最后是统计分析，例如对缺陷分布进行分析。

RUP 测试流程框架如图 3-6 所示，软件测试团队可以以它为基础，根据业务发展的实际要求，定制符合团队使用的软件测试流程。

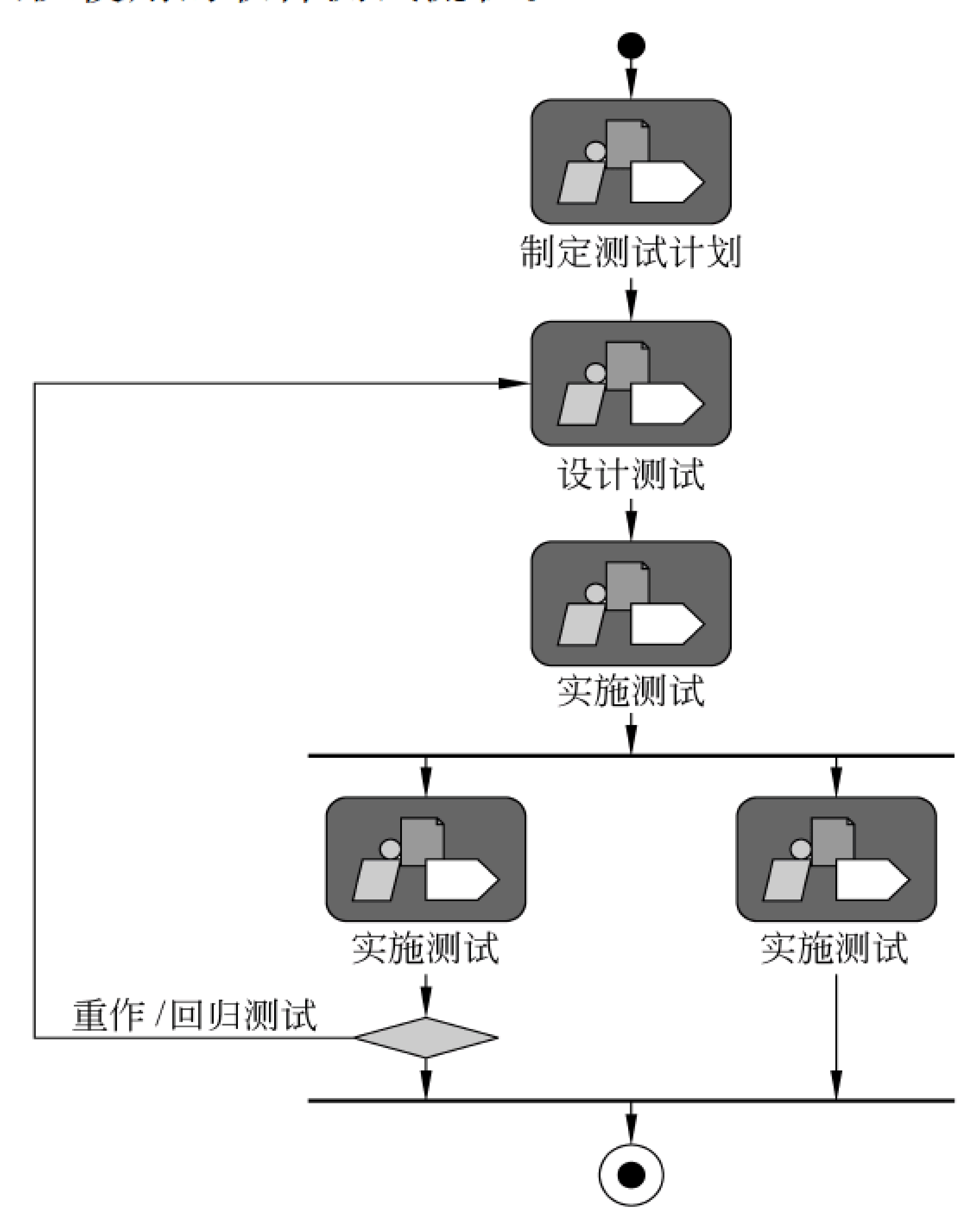


图 3-6 RUP 软件测试流程

名词解释

（1）测试用例

RUP给出的定义是测试用例记录需要在受测试系统中进行验证的行为单元。测试用例记录始终保持至少与一个测试计划记录相关。简单地讲，为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以便测试某个程序路径或核实是否满足某个特定需求。这些特殊目标可以是验证一个特定的程序路径或核实是否符合特定需求。因此测试用例文档指对一项特定的软件产品进行测试任务的描述，体现测试方案、方法、技术和策略。内容包括测试目标、测试环境、输入数据、测试步骤、预期结果、测试脚本等。

影响软件测试的因素很多，例如软件本身的复杂程度、开发人员（包括分析、设计、编程和测试的人员）的素质、测试方法和技术的运用等。因为有些因素是客观存在的，无法避免。有些因素则是波动的、不稳定的，例如开发队伍是流动的，有经验的人走了，新人不断补充进来；一个具体的人工作也受情绪等影响，等等。如何保障软件测试质量的稳定？有了测试用例，无论是谁来测试，参照测试用例实施，都能保障测试的质量。可以把人为因素的影响减少到最小。即便最初的测试用例考虑不周全，随着测试的进行和软件版本更新，也将日趋完善。

因此测试用例的设计和编制是软件测试活动中最重要的。测试用例是测试工作的指导，是软件测试必须遵守的准则。更是软件测试质量稳定的根本保障。

（2）测试需求

测试需求定义了“做什么”。这个概念比较简单，例如要对一个哈希表的插入操作进行测试，首先要考虑做什么，会出现什么情况，把所有的情况都考虑到，这样就得到如下所有的测试需求：

① 插入一个新的条目；

② 插入失败——条目已经存在；

③ 插入失败——表已满；

④ 哈希表在插入前为空。

这些就是测试需求，需要注意的是测试需求与测试用例是有区别的，测试用例是选择满足这些测试需求的输入值，即测试数据。一个简单的测试用例可能会同时满足好几个测试需求。一个用例能同时满足好几个测试需求当然是最理想的情况，但是这样做的代价较高。另外一种方法是为每一个测试需求设计一个单独的测试用例，就可以不必考虑那些复杂的测试用例，但是这些相对简单的测试用例发现缺陷的能力就会有所下降。

换句话说，测试需求并没有指出具体的数值或数据，如哈希表插入什么数据，而测试用例对被插入元素进行描述。

（3）测试计划

测试计划表示预计要执行的测试的主要分组。它可包含对关联的子测试用例记录的引用，或对进一步指定相关测试区域的其他测试计划的引用。测试计划记录提供项目中其他测试资产的组织结构。简单地说，测试计划就是描述了要进行的测试活动的范围、方法、资源和进度的文档。它确定测试项、被测特性、测试任务、谁执行任务、各种可能的风险。测试计划可以有效预防计划的风险，保障计划的顺利实施。

在 RUP 中对测试流程的描述为：软件测试工作要通过制定测试计划、设计测试、实施测试、执行测试、评估测试几个阶段来完成。每个测试环节的具体阐述如下：

① 制定测试计划需要整理测试需求，目的是确定和描述要实施和执行的测试。这是通过生成包含测试需求和测试策略的测试计划来完成的。可以制定一个单独的测试计划，用于描述所有要实施和执行的不同测试类型，也可以为每种测试类型制定一个测试计划。两者都是可以采用的方法。

该环节由测试设计员负责执行。无论所要开发的系统的大小如何、复杂程度如何，对其进行尽量完整的测试是十分重要的，所以一般说来，执行测试计划涉及确定能够提供最佳投资回报率的测试用例、测试过程与测试组件。此时主要是把重点放在测试最重要的用例和那些与最高风险有关的非功能性需求方面。

② 设计测试要设计测试用例和测试过程，保证测试用例完全覆盖测试需求，目的是确定、描述和生成测试过程和测试用例。该环节同样由测试设计员负责执行。

设计测试涉及设计不同等级的测试，以便针对内部发布的每个系统版本进行测试，并给出执行这些测试所应遵循的程序。测试设计员应尽可能从回归测试重用的角度来设计集成测试用例和系统测试用例。回归测试倾向于把重点放在重新测试已更改的代码或重新测试风险较大的用例实现上。

③ 实施测试要根据测试用例实现具体的自动化脚本或手工操作步骤，目的是实施（记录、生成或编写）设计测试中定义的测试过程。输出工件是测试过程的计算机可读版本，称为测试脚本。测试脚本的生成可以在测试自动化工具环境中或编程环境中完成。另外，还需要用适当的工具和方法创建在执行测试脚本时所需要的外部数据集。该环节由测试设计员负责执行。

④ 执行测试则通过自动化测试工具或手工来执行那些自动化或手工脚本，目的是确保整个系统按既定意图运行。系统集成员在各迭代中编译并链接系统。每一迭代都需要测试增加的功能，并重复执行以前版本测试过的所有测试用例（回归测试）。本活动涉及为系统的每个工作版本执行手动集成测试与自动化集成测试，并报告缺陷。该环节由测试人员负责执行。

⑤ 评估测试要对软件的质量和测试工作自身的质量做出一个客观的评价，目的是生成并交付测试评估摘要。这是通过复审并评估测试结果，确定并记录变更请求，以及计算主要测试评测方法来完成的。测试评估摘要以组织有序的格式提供测试结果和主要测试评测方法，用于评估测试对象和测试流程的质量。该环节由测试设计员负责执行。

这样的过程是需求管理，可通过自动化测试工具把需求细化到每一个用例，甚至一个需求用例对应好几个测试用例，包括测试分支或正反向分支等。经常会有这样的情况发生，例如测试用户不同有不同的情况出现，需求和测试用例对不上，客户发现测试人员没发现的问题。这个时候可利用测试工具来进行循环测试，每一个迭代产生一部分版本，由于更改可能带来新的 Bug，因此要做回归测试。

案例分析：中兴软件外包测试流程

如果竞标成功，项目就要开始启动。中兴方会提供一份 CRS（客户需求）和 SOW（工作任务书），中兴方派人过来进行需求培训，这时该项目的测试组长也要参与到项目需求

的培训和评审,也就是测试工作从需求开始介入。项目经理编写《项目计划》,开发人员产出《SRS》,这时测试组长就要根据 SOW 开始编写《测试计划》,其中包括人员、软件硬件资源、测试点、集成顺序、进度安排和风险识别等内容。

《测试计划》编写完成后需要进行评审,参与人员有项目经理、测试经理和中兴方人员,测试组长需要根据评审意见修改《测试计划》,并上传到 VSS 上,由配置管理员管理。待开发人员把《SRS》归纳好并打了基线,测试组长开始组织测试成员编写《测试方案》,测试方案要求根据《SRS》上的每个需求点设计出包括需求点简介,测试思路和详细测试方法三部分的方案。《测试方案》编写完成后也需要进行评审,评审人员包括项目经理、开发人员、测试经理、测试组长、测试成员和中兴方。如果中兴方不在公司,就需要测试组长把《测试方案》发送给中兴进行评审,并返回评审结果。测试组长组织测试成员修改测试方案,直到中兴方评审通过后才进入下个阶段——编写测试用例。

测试用例是根据《测试方案》来编写的,通过《测试方案》阶段,测试人员对整个系统需求有了详细的理解。这时开始编写用例才能保证用例的可执行和对需求的覆盖。测试用例包括测试项、用例级别、预置条件、操作步骤和预期结果。其中操作步骤和预期结果需要编写详细和明确。测试用例应该覆盖测试方案,而测试方案又覆盖了测试需求点,这样才能保证客户需求不遗漏。同样,测试用例也需要通过开发人员、测试人员和中兴方评审,测试组长也要组织测试人员对测试用例进行修改,直到中兴方评审通过。

在编写测试用例阶段,开发人员基本完成代码的编写,同时完成单元测试。中兴的外包项目一般是一次性集成,所以软件转到测试部门后直接进行系统测试。测试部门对刚转过来的测试版本进行预测试,如果软件未实现 CheckList 清单上的 10%,测试部门会把该版本打回。否则,软件转到测试部门进行系统测试。根据《测试计划》进度安排,测试组长进行多轮次的测试,每轮测试完成后,测试组长需要编写测试报告,其中包括用例执行通过情况、缺陷分布情况、缺陷产生原因、测试中的风险等。这时测试人员就修改增加测试用例。待到开发修改完 Bug 并转来新的测试版本,测试部门开始进行第二轮的系统测试,首先回归完问题单,再继续进行测试,编写第二轮的测试报告,如此循环下去,直到系统测试结束。在系统测试期间,测试人员还需要编写验收手册、验收用例和资料测试用例等。

完成系统测试后,软件就开始转到中兴进行验收测试,其中大概测试半个月,一般会要求测试部门派人到中兴方进行协助测试,并发回问题单给公司开发人员修改。如果验收发现的缺陷率在 SOW 规定的范围内,那么验收成功,中兴方付钱给公司,项目结束。如果超过规定的缺陷率,那么公司要罚款,整个项目组成员(包括开发和测试)也相应要接受处罚。

3.2.2 RUP 软件测试评测方法

软件测试的主要评测方法包括测试覆盖和质量评测。测试覆盖是对测试完全程度的评测,它是由测试需求和测试用例的覆盖或已执行代码的覆盖表示的。质量评测是对测试对象(系统或测试的应用程序)的可靠性、稳定性以及性能的评测,它建立在对测试结果

的评估和对测试过程中确定的变更请求(即缺陷)分析的基础上。

1. 覆盖评测

覆盖指标提供了“测试的完全程度如何?”这一问题的答案。最常用的覆盖评测是基于需求的测试覆盖和基于代码的测试覆盖。简而言之,测试覆盖是就需求(基于需求的)或代码的设计/实施标准(基于代码的)而言的完全程度的随机评测,如用例的核实(基于需求的)或所有代码行的执行(基于代码的)。

(1) 基于需求的测试覆盖

基于需求的测试覆盖在测试生命周期中要评测多次,并在测试生命周期的里程碑处提供测试覆盖的标识(如已计划的、已实施的、已执行的和成功的测试覆盖)。

测试覆盖通过以下公式计算:

$$\text{测试覆盖} = T^{(p,i,x,s)}/\text{RFT}$$

其中,T 是用测试过程或测试用例表示的测试数(已计划的、已实施的或成功的),RFT 是测试需求(requirement for test) 的总数。

(2) 基于代码的测试覆盖

基于代码的测试覆盖评测测试过程中已经执行的代码的多少,与之相对的是要执行的剩余代码的多少。代码覆盖可以建立在控制流(语句、分支或路径)或数据流的基础上。基于代码的测试覆盖通过以下公式计算:

$$\text{测试覆盖} = I^{e}/\text{TIIC}$$

其中,I^{e} 是用代码语句、代码分支、代码路径、数据状态判定点或数据元素名表示的已执行项目数,TIIC(total number of items in the code) 是代码中的项目总数。

2. 质量评测

测试覆盖的评估提供对测试完全程度的评测,对在测试过程中已发现缺陷的评估提供了最佳的软件质量指标。因为质量是软件与需求相符程度的指标,所以在这种环境中,缺陷被标识为一种更改请求,该更改请求中的测试对象与需求不符。

(1) 缺陷报告

一般可以将缺陷计数作为时间的函数来报告,即创建缺陷趋势图或报告;也可以将缺陷计数作为一个或多个缺陷参数的函数来报告,例如作为缺陷密度报告中采用的严重性或状态参数的函数。这些分析类型分别为揭示软件可靠性的缺陷趋势或缺陷分布提供了判断依据。

RUP 以三类形式的报告提供缺陷评估:

① 缺陷分布(密度)报告允许将缺陷计数作为一个或多个缺陷参数的函数来显示。用于描述当前软件的质量状态。

② 缺陷龄期报告是一种特殊类型的缺陷分布报告。缺陷龄期报告显示缺陷处于特定状态下的时间长短。一般用它来表示研发团队对质量的反应能力。

③ 缺陷趋势报告按状态将缺陷计数作为时间的函数显示。趋势报告可以用来标识软件质量变化的趋势。

(2) 性能评测

评估测试对象的性能行为时,可以使用多种评测,这些评测侧重于获取与行为相关的数据,如响应时间、计时配置文件、执行流、操作可靠性和限制。这些评测主要在“评估测试”活动中进行评估,但是也可以在“执行测试”活动中使用性能评测评估测试进度和状态。

主要的性能评测包括:

① 动态监测——在测试执行过程中,实时获取并显示正在执行的各测试脚本的状态。

② 响应时间/吞吐量——测试对象针对特定主角和/或用例的响应时间或吞吐量的评测。

③ 百分位报告——数据已收集值的百分位评测/计算。

④ 比较报告——代表不同测试执行情况的两个(或多个)数据集之间的差异或趋势。

3.3 质 量 保 证

国内很多软件公司专门设置了软件测试部门,负责软件测试工作。但是在很多大型公司例如摩托罗拉、爱立信等公司里,软件测试是由质量保证(QA)部门来负责的,这些部门的工作主要是放在过程管理上,对制造过程中的每一道工序都要进行质量控制,测试工作只是其中的一个环节。他们认为质量不是依赖于某个或某几个高手,而是依赖于整个过程。好的过程是好产品的必备条件,而且为了较好地开展软件质量保证工作,软件质量保证部门是独立的,与项目经理没有任何行政隶属关系,同时 QA 部门也不能承担本项目中除软件质量保证外的其他任何工作,以确保其独立性。

质量保证案例

联想公司的软件产品是通过“质量保证检查表”来实现的。针对每个软件工程活动与软件工作产品,都有一个软件质量保证检查表。软件质量保证人员可以根据检查表来判断当前的软件工程活动是否存在偏离以及软件工作产品是否符合要求。软件工程活动是否有所偏离主要看软件工程活动的进入准则是否达到,输入是否正确,执行任务是否符合要求,结束时是否符合完成准则,以及是否具有合乎要求的输出。在审计软件工作产品时,软件质量保证人员主要审计软件产品是否符合规程、标准等要求,一般不考虑技术问题。

联想所有软件开发人员都接受过软件质量保证方面培训,了解软件质量保证的目的、工作方式及其他相关内容。联想认为只有所有的人都认识到软件质量保证工作的意义,这项工作才能真正很好地开展起来,质量才能从根本上得到保证。

3.3.1 过程质量保证

软件开发过程质量是指为了生成工件而对可接受流程(包括质量评测和质量标准)的

实施和遵守程度。软件生产的过程质量与本章开头的汽车例子类似，体现在三个层次：

一是产品本身和用来生产、组装软件产品的零部件质量，包括用来进行软件开发或在软件开发过程中产生的代码、文档、模型和可执行系统等工件；

二是软件开发活动本身对标准化软件开发过程的遵守程度，主要体现在软件开发过程的标准化、流程化、自动化程度和团队基本协作平台的效率；

三是用来对整个软件产品进行验收的评测手段，这是被业界广泛认可和接受的方法。

一个软件生产企业的过程质量，一般可以用它的软件过程成熟度等级(例如 CMM/CMMI 级别)来决定。高成熟度等级也正是印度的软件公司能够获取很多外包项目的重要原因。但应该更清醒地看到：真正保证软件质量的不是 CMM/CMMI 的一纸评估报告，而是软件生产过程本身的成熟度，包括我们赖以达到成熟等级的方法、工具和软件开发平台。可喜的是国内越来越多的企业已认识到这一点，把更多的工夫花在使用合适的方法、采用恰当的工具和平台，切实提高软件生产过程的成熟度。

质量控制从需求开始，需求分析和需求管理方面的技巧和技术方法有很多，它们都从需求方面来保证软件的质量；到了设计阶段，也有很多成熟的设计方法，例如可视化建模，基于组件的架构设计等；再到实现，到测试等方面，都有很多的方法和技巧来提高软件的质量，例如面向对象技术等。这里面每一个环节的目的都是为了提高整个软件开发的质量。

3.3.2 质量保证与 RUP 的关系

其实 RUP 整个流程都在讲软件质量保证。RUP 里定义了一个软件生命周期模型，分成四个阶段：初始阶段、细化阶段、构造阶段、交付阶段。每个阶段有不同的侧重点，通过多次的迭代，每次迭代里面都要做质量控制。只不过 RUP 专门侧重于从软件开发的整个生命周期来保证软件质量，所以对软件开发商特别适合。而其他模型侧重点在于其他环节，例如 ISO9000 质量管理体系就适合用在传统制造业。

RUP 是一个成熟的软件工程过程，为了保证软件过程质量，主要工作有建立有效的工作过程，提高团队的生产效率，控制开发过程中的风险，保证软件开发进度并提高软件产品质量。同时通过为所有重要的开发活动提供全面的指南、模板和示例，使整个软件开发团队能够有效共享成功经验，提高团队效率。

RUP 里定义了每一个角色，每一个角色跟 QA 人员都有关系。例如 RUP 里分了九个规程，流程工程师属于环境规程里边的角色，项目经理是项目管理规程里边的角色。每一个规程其实就是一类开发活动，其中的角色和他们所产生的工件集合是一个分类。可以把项目经理相关的工作，所涉及的工件，例如软件开发计划、风险管理计划、质量保证计划等都放在项目经理规程里面，这样 QA 人员跟项目经理的关系就是去检查项目经理在这个岗位上所做的职责是否到位，是不是跟流程相符合。其他角色也一样，例如检查测试人员的工作就看测试人员有没有根据规定把缺陷按正确的测试流程汇报，发现缺陷之后是否能够得到改正，并作一个复审，回归测试有没有考虑测试的完备性等问题，有没有做好具体的工作。QA 人员和整个项目团队在工作中的关系就是看每一个角色是不是很好

地完成了自身角色所应该完成的开发任务。依据标准就是这个组织的流程，流程是保证质量的一个重要依据。

QA 人员判断其工作效果和质量最直接的依据就是 RUP 工件。可以去检查这些工件，可以根据检查的结果来判断角色是否达到了要求。在检查结果中，必然涉及如何在同一标准下对结果进行评判的问题，也就是说开发团队应该采用统一的开发方法和流程。不然的话，每一个开发团队各自采用不同的开发流程，流程工程师就没办法去评价，没有一个可对照的标准，没有可比性。另外团队采用统一的工具和开发平台，会帮助自动收集很多的信息。例如 IBM 公司的 Tester Manage 测试管理平台可以帮助了解项目进度，以及项目进行过程中产生的各种结果，包括测试的报告等，这些都要有一个统一的标准。就像现在的航空公司都会选择相同飞机制造厂商的机型，目的就是要降低维护的成本。因为机型统一，就比较好进行管理。在一个软件企业，内部采用统一的软件开发平台，也能有助于企业判断项目的情况，判断的方法也会相对简单，还可降低工作量。

质量保证角色案例

RUP 里面有一个角色叫 Process Engineer（过程工程师），在中兴属于中兴 QA 部门，他的工作就是负责制定整个软件开发的流程。因为真正要保证质量的话，不能仅靠测试来保证，必须在整个开发流程的各个环节都要做得很好，才能够真正地提升软件的质量。所以制定一个好的流程，在某种程度上决定软件的开发能不能按时交货，能否保证软件质量。这个流程就是由 QA 部门来制定的。

中兴 QA 部门还有另外一个职责，就是保证整个研发团队能够严格按照这个流程来运作。在项目到达每一个里程碑的时候，中兴 QA 部门的 QA 经理就会介入，对项目做一个审核，检查前一阶段的工作是否按照公司制定的流程来运作。看看该有的工件是不是都有了，该做的步骤是不是都做了，开发团队要证明给 QA 人员看。只有过了这一关，QA 部门才会同意开发团队可以进行下一步的工作。所以从广义上理解，软件质量保证是针对整个软件开发流程的，是关系怎样在软件开发生命周期中来保证软件的质量。

3.3.3 RUP 全过程质量保证思想

RUP 把整个软件开发过程分解成：业务建模、需求管理、分析设计、实施、测试、部署、配置与变更管理、项目管理和环境等九个核心工作规程。每个核心工作规程由多个详细工作流程组成。基于人类对软件工作过程最原始的感受，RUP 使用角色、活动和作为输入输出的工件来组织每个详细工作流程，实现软件开发组织内部人、资源和流程的融合。RUP 通过建立完整的软件开发过程，使得产品的质量由项目团队的每个成员共同负责，具体体现在：

① 每个角色承担相应的质量任务。

- 每个活动产生合格的工件；
- 为每个工件建立指南、模板和检查点。

② 每个工作流程设定相应的工作指南和检查点。

在 RUP 中，整个软件开发过程如图 3-7 所示，它以指定的工件为输入，通过软件开发

角色和标准化的软件开发活动，生产出满足质量要求的输出工件。为确保每个工作环节的有效执行和每个工作环节产生的工件质量，RUP 为主要工作流程提供了对应的工作指南和检查点，为每个工件建立指南、模板和检查点，从而保证了软件开发的过程质量。

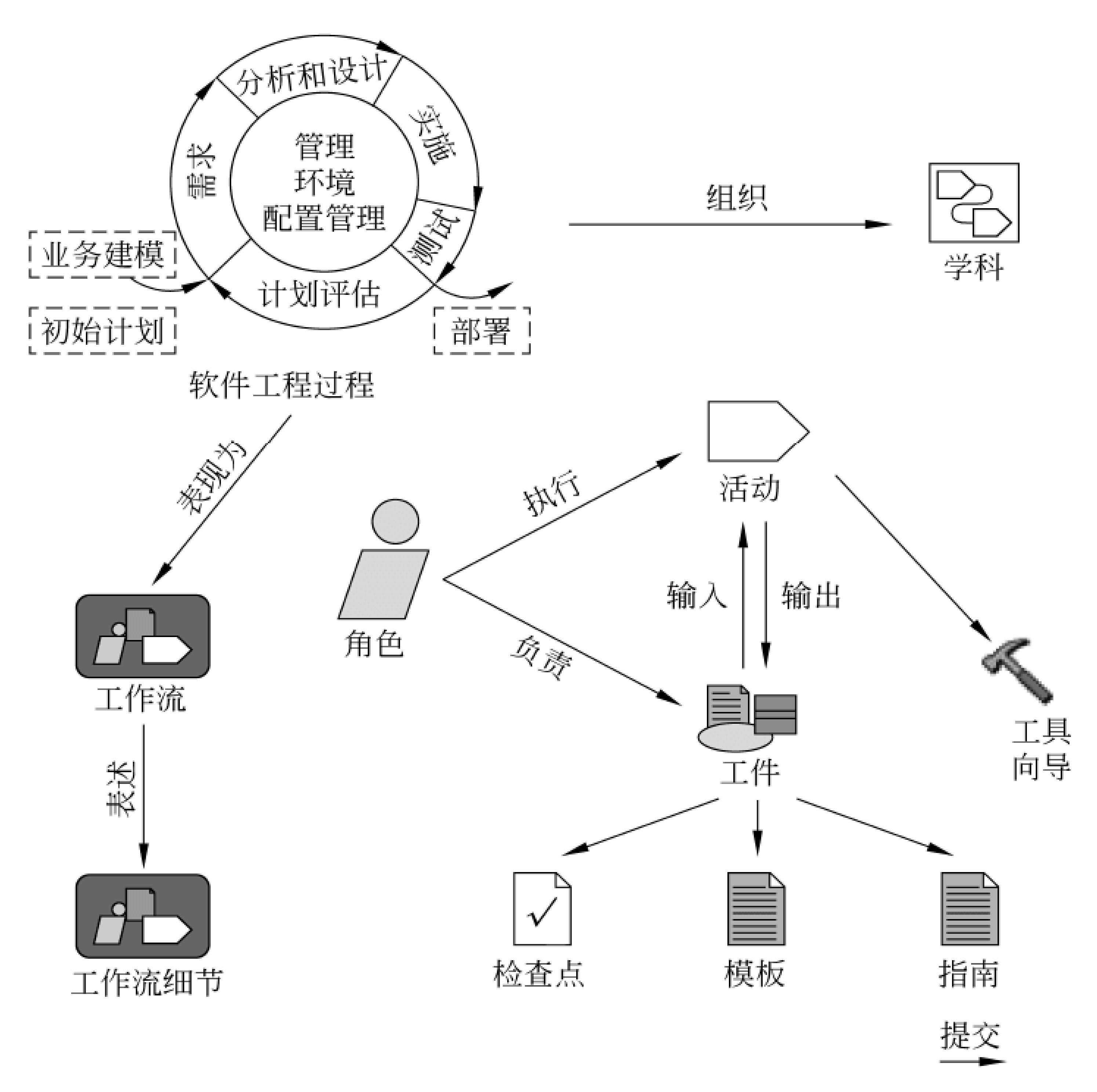


图 3-7　RUP 开发过程

整个软件开发过程都要引入测试的理念，要同各个团队打交道，测试团队并不只是进入最终代码一级才有的工作，测试团队要跟很多部门沟通、交流，这些部门有市场部门、沟通渠道、对外交流、项目管理、IT 部门。

传统的测试案例

如某公司经常要测试其架设的网络是否正常，运行是否良好。过去开展测试网络工作，一般要有 IT 部门的介入，开始部署系列工具，应用架构设计方法等，这一系列都需要公司高层来指挥实施，同时制定不同阶段不同的协作机制。

为此需要划分不同测试阶段，从最早项目的起始阶段，需求分析，到设计阶段，项目分析，再到系统测试、安装和维护阶段，传统的测试安排在设计的后期，因此测试人员会介入或了解一些设计文档，到编码后期就开始测试工作。越到后面发现问题成本越高，而且传统观点认为只能在测试阶段发现很多 Bug，而实际上这些 Bug 很多是在设计这一块出现的，并不是在编码。设计的理念或需求分析就有偏差，可以看出传统测试存在不够科学的问题，与前面讲述 RUP 理念相悖。

RUP 测试模型里，定义阶段一般要定义项目应达到什么目标，项目范围多大，什么阶段产出什么样的产出物，这个时候要做好测试的准备工作，测试的计划就会开始启动。

测试准备开始后，在软件设计阶段和代码生成阶段，单元测试、集成测试、系统测试等测试都会开始引入到这些阶段。此外还有测试软件的配置管理，测试环境的建立。这样的测试准备贯穿于软件生命周期，即需求阶段、定义阶段，并一直延续到交付维护阶段的全部过程。

由此可见，测试一直伴随着软件生命周期，不断地进行迭代，这样的测试是在不断地循环。

问题：测试为什么人服务

很多人认为测试只是为开发团队服务，这个认识肯定不完全、不准确。通常看到测试团队与开发团队协作非常紧密。事实上不仅需要跟开发团队合作、服务，还要跟需求的业务分析部门，甚至跟后面的运营部门合作或服务。产品上线了，或者递交到外部，也需要测试服务。我们更多的是要以整合业务的角度来看待测试，而且这不仅是如何看待的问题，也是对测试理念的认识。例如在业务驱动的软件开发测试生命周期里，有不同角色参与进来，即最终用户，上层的管理层，还有测试人员、开发人员、架构设计师。测试工作是循环性的，它跨平台、跨部门，渗透在各个部门、各个阶段。这就是 IBM 公司提出的业务驱动开发思想。

3.3.4 软件工程成功经验铸就软件质量

激烈的市场竞争催生出高质量的软件。同时，软件行业经过几十年的发展，软件生产工艺、软件开发方法和工具都大大进步、日趋成熟，这一切使软件开发质量越来越好。RUP 以迭代式的软件开发、架构为核心的软件开发、用例驱动的软件开发和风险驱动的软件开发为特色，集中体现了软件工程的六个成功经验，通过它们共同铸就了高品质软件。

与汽车生产过程相比，企业级项目管理平台和团队统一平台就好比汽车生产过程的生产线，统一了整个软件的开发活动和管理活动，而其他各种根据软件开发角色组织的开发工具，就好比生产线上的各种智能机床，大大提高软件生产过程的生产力和质量保证。如果说福特使用的生产线技术实现了汽车行业的规模化生产，那么按照 RUP 打造的软件开发平台无疑为软件的规模化生产揭开了序幕。

一个好的软件开发平台为软件生产提供了涵盖分析员、架构师、设计员在内的全生命周期的质量保证，而不仅是对测试人员的质量解决方案。在 RUP 质量保证思想的指导下，软件开发平台要坚持软件质量从头抓起的宗旨，使用业务建模工具和可视化建模技术，准确描述企业的业务流程，模拟企业的业务执行过程，帮助企业找出业务流程中存在的问题，优化业务流程。系统分析人员借助业务模型，可以准确理解企业需求，解决企业真正需要解决的问题，正确构建企业需要的系统。

3.4 测试团队与角色

测试团队可以是测试部，也可以是测试组。公司规模决定了测试团队的大小和组织形式。测试团队建设需要执行两个原则：第一，测试团队必须独立于开发团队之外，而不

是附属于开发团队,实现测试的独立性和公正性;第二,测试团队必须具有明确的工作目标,即发现和报告软件缺陷,推动和确认缺陷修正,协助软件开发的过程改进,提高软件整体质量。

案例

Microsoft 公司起步初期,许多软件产品出现了“Bug”。例如 1981 年与 IBM PC 一起推出的 Basic 软件,用户在用“.1”(或者其他数字)除以 10 时,就会出错,由此激起了许多采用 Microsoft 操作系统的 PC 厂商的极大不满,而且很多个人用户也纷纷投诉。

Microsoft 公司高层领导觉得很有必要引进更好的测试与质量控制方法。但是遭到很多开发人员甚至一些高层经理的反对,他们认为开发人员可以自己测试产品,或者让学生和行政人员来协助测试就可以了。

1984 年 Microsoft 特地请 Arthur Anderson 咨询公司测试 Mac 机的 Multiplan(电子表格软件)。但是该公司没有能力执行全面的软件测试。结果漏测的一个 Bug 迫使 Microsoft 公司为它的 2 万多名用户免费提供更新版本,损失达 20 万美元。

痛定思痛后,Microsoft 公司得出一个结论:如果再不成立独立的测试部门,软件产品就不可能达到更高的质量标准。

软件测试团队根据规模可以设置多个职位,每个职位具有明确的岗位职责,例如,测试部门经理、测试项目经理、测试组长、测试架构师、高级测试工程师、测试工程师等。对于刚刚成立的测试团队,可以一个人兼任多个职位,完成多项测试任务。测试人员的总数应该与开发人员相适应,最好在 1∶1 到 1∶2 之间。

团队中的工作人员扮演着不同的角色,每个角色都有明确的职责、相关技能要求。对于每个项目,所有的角色都需要进行合理的安排(包括人员的数量和人员本身的素质),因此就有可能存在一个人扮演同一项目的不同角色或在多个项目中扮演相同(甚至不同)的角色。

3.4.1 RUP 中测试角色

1. 测试经理(Test Manager)

测试经理是负责项目测试任务,以确保测试活动成功的角色。

(1) 职责

① 协商测试工作的目标与提交的成果,管理测试活动的范围,并据此制定测试计划。

② 为测试活动分配人力资源和获取测试设施。

③ 监督项目测试活动的进度和效果。

④ 解决阻碍测试开展的矛盾和问题。

⑤ 通过发现重要的缺陷来推进项目产品的质量水平。

⑥ 关注软件开发过程并推动改善工件(需求、代码等)的可测试性。

(2) 专业技能

① 具备软件开发过程各个方面的基本知识。

② 拥有测试方法、技术和工具等广泛的经验。

③ 掌握计划和管理的技能。

④ 熟悉被测试系统领域的相关知识。

⑤ 拥有编程经验。

活动包括确定测试任务、识别测试动因、获取测试承诺、评估和推进产品质量、评估和改进测试活动。

工件有测试计划、变更请求、事项列表、测试评估总结。

2. 测试分析员(Test Analyst)

测试分析员是负责识别和定义所需测试,监督具体测试进展和成果的角色。

(1)职责

① 识别将通过测试来验证的测试对象条目。

② 定义合适的测试要求和相关的测试数据。

③ 收集和管理测试数据。

④ 分析各测试周期的结果。

(2)专业技能

① 具备软件开发过程各个方面的基本知识。

② 拥有良好的分析技能。

③ 关注细节并且坚韧不拔。

④ 对软件常见的失效与错误有充分理解。

⑤ 拥有测试方法、技术和工具等广泛的经验。

⑥ 熟悉被测试系统领域的相关知识。

⑦ 拥有测试经验。

⑧ 可以由需求阐释员兼任,方便按照用例编制测试用例。

活动有识别测试对象、确定测试思路、定义测试细节、确定评估和跟踪要求、判断测试结果、验证各构造版本中的变更。

工件有测试计划、测试评估总结、变更请求、测试指南、测试思路列表、测试用例、测试数据、测试结果记录、工作负载分析模型。

3. 测试设计员(Test Designer)

测试设计员是负责针对测试目标设计测试途径以确保测试被成功实施的角色。

(1)职责

① 确定并描述相应的测试技术。

② 确定相应的测试支持工具。

③ 定义并维护测试自动化架构。

④ 详述和验证需要的测试环境配置。

⑤ 验证与评估测试途径。

（2）专业技能

① 具备软件开发过程各个方面的基本知识。

② 拥有验证测试成果的经验。

③ 具备诊断和解决调试问题的技能。

④ 拥有硬件与软件安装、配置等的广博知识。

⑤ 拥有使用自动化测试工具的成功经验。

⑥ 对软件常见的失效与错误有充分理解。

⑦ 深入掌握被测试系统领域的相关知识。

⑧ 拥有编程经验。

⑨ 具备开发团队领导和软件设计的技能。

⑩ 可以由软件架构师充当本角色。

活动包括定义测试途径、确定测试机制、定义测试环境配置、组织测试实施元素、定义测试元素。

工件有测试计划、测试脚本、测试自动化构架、测试界面规格、测试环境配置、测试套件。

4. 测试员（Tester）

测试员是负责遵照设计的测试途径负责实施测试的角色。

（1）职责

① 为给定的测试确定最合适的实施途径。

② 实施各个测试。

③ 设置并执行测试。

④ 记录测试结果并验证测试的执行。

⑤ 分析执行遇到的错误并从中恢复。

（2）专业技能

① 具备软件开发过程各个方面的基本知识。

② 接受过使用相应自动化测试工具的培训。

③ 拥有使用自动化测试工具的经验。

④ 具备诊断和调试的技能。

⑤ 拥有编程经验。

⑥ 可以由测试分析员充当本角色。

活动包括实施测试、实施测试套件、执行测试套件、分析测试失败。

工件有测试套件、测试脚本、测试记录、变更请求。

3.4.2 RUP 测试制品

RUP 中制定了很多测试制品，包括：

1. 测试评估总结(Test Evaluation Summary)

测试评估总结用于整理并展示复审与评估的测试结果及测试的主要评测方法。此外,测试评估摘要包括测试员和测试设计员对这方面信息的评估,还包括他们对将来工作的建议。它是测试人员与其他团队成员沟通时最重要的制品。

主要内容包括测试结果摘要、基于需求的测试覆盖、基于代码的测试覆盖、缺陷分析及建议措施等。

2. 测试计划(Test Plan)

从前面定义知道测试计划包含项目范围内的测试目的和测试目标的有关信息。此外,测试计划确定了实施和执行测试时使用的策略,同时还确定了所需资源。在项目的一开始就应创建最初的测试计划,该计划称为"主测试计划"。主测试计划仅提供计划的全部测试工作的概述。随着每次迭代的筹划,将创建一个或多个更精确的"迭代测试计划",其中包含与特定测试需求、测试策略、资源等有关的更精确的信息,这些信息都对应于每次具体迭代。所有测试计划内容都建立在测试计划模板的基础上。

3. 测试脚本(Test Script)

测试脚本是自动执行测试过程(或部分测试过程)的计算机可读指令。测试脚本可以被创建(记录),或使用测试自动化工具自动生成,或用编程语言编程来完成,也可综合这三种方法来完成。

4. 测试用例(Test Case)

从前面知道测试用例是为特定目标开发的测试输入、执行条件和预期结果的集合。测试用例比测试脚本更抽象。另外,它还规定了开始条件、测试过程的条件和结束条件。

测试用例的目的是确定并传达一些条件,这些条件将在测试中执行,并且是核实实施产品需求(用例、性能特征等)是否成功和能否接受所必需的条件。测试用例反映了一种测试覆盖(基于需求的测试覆盖)评测方法,这是因为每个测试用例都可追踪到至少一个测试需求,而这些需求则反映出产品的需求。

5. 测试组(Test Suite)

测试组是一组相关的测试,它们在一起执行可以给出关于某个方面的更好的评估。一个测试组实现了一个或多个包含测试脚本和测试数据的测试思想或测试用例。

如果只想运行一个测试用例,测试员可以从测试计划中直接运行。对于快速执行少量的测试用例来说,这种方法很好。如果要执行的测试用例比较多,就有点力不从心了,测试组就是一个很好的解决方法,创建一个包含要执行的测试用例的集合(suite)。集合可以被存储,并在任何时候再次执行。集合也能提供一些附加的功能,如分发测试到远程工作站执行,还允许测试员使用一些功能如随机执行、同步、延迟、分组等。

3.5　RUP 四级测试

RUP 确定了四级测试：单元测试、集成测试、系统测试和验收测试。这些测试级别可以是并列的，也可以是递进的，这取决于主测试计划（在项目级）和迭代测试计划（在迭代级）。

3.5.1　主测试计划和迭代测试计划

RUP 对整个项目使用一个主测试计划，对每次迭代使用一个迭代测试计划。测试经理先草拟主测试计划。尽管是从系统测试开始的，但为了排列这些测试级别，在这两种情况中，计划都有可能包括系统测试之外的其他测试级别。这两个计划在内容上有大量的相似之处，只是范围和详细的程度有所区别。一个选择是将迭代测试计划与迭代计划集成在一起。在这种情况下，测试的贡献主要是在于指出了确定和执行测试用例所依据的需求。与此迭代相关的其他测试活动也被提出来了，例如适当的测试工具的选择或创建明确的指导方针。

3.5.2　单元测试

单元测试用于软件的最小可测试单元。单元测试强调内部结构，例如逻辑和数据流，以及单元的功能和外部可见行为测试。

RUP 中的单元测试是实现人员的明确任务，在对于新的或变更单元的每个迭代中，都是由实现人员来执行实现测试组件和子系统，以及执行单元测试这些活动。这就使得测试成为与此角色相关联活动的主要部分。

尽管 RUP 包含了许多关于单元测试的指导方针，但是并没有命名一个项目特定指导方针需要在其中被固化的工件。这就和用例设计和用例建模的指南形成了对比。在理论上，单元测试的指导原则被包括在单元测试指南中，指南的开发和执行是过程工程师的共有责任，并且应当与主测试计划一致。

3.5.3　集成测试

集成测试取决于当软件组件在被合并起来执行一个用例时，是否功能正确。开发人员将自己单元测试过的组件提交到集成人员那里，集成人员将这些单元合并成一个中间构造。这种一步步的组件集成发生在从下至上的方式中，并且按照集成构建计划规定的顺序进行。在每一步之后，集成人员组成一个中间构造，被提交用于执行集成测试。主要目标是确定这些组件与已经集成组件之间的兼容性。结果常常会执行集成构建计划的一个子集。这种一步步的方法考虑到足够的问题隔离和分析。

RUP 规定，集成测试由测试角色的人员来执行，也可以合并在集成员角色下面。实际上这些角色为了效率最大化，可以由相同的人员来承担。这意味着，集成人员也可以执行作为其活动主要内容的集成测试。

关于指南和工具，应用于集成测试与单元测试是一样的。主要的差别是，集成测试的指南不会自动成为编程指南的一部分，但会是测试指南的一部分，或遵循集成构建计划。测试经理需要与其他工程师进行一些调整。

3.5.4 系统测试

当软件功能成为一个系统时，在不同组件的集成（测试）之后，就开始执行系统测试。多个构建可以在一个迭代中交付。通常每个构建都要进行一次系统测试，除非集成测试计划另有规定。主测试计划和更具体的迭代测试计划，需要简要说明哪个构建需要被测试。

RUP 确定了系统测试的测试流程，如图 3-8 所描述的，缺省规程、活动、工件、角色等都在此流程中进行了详细描述。

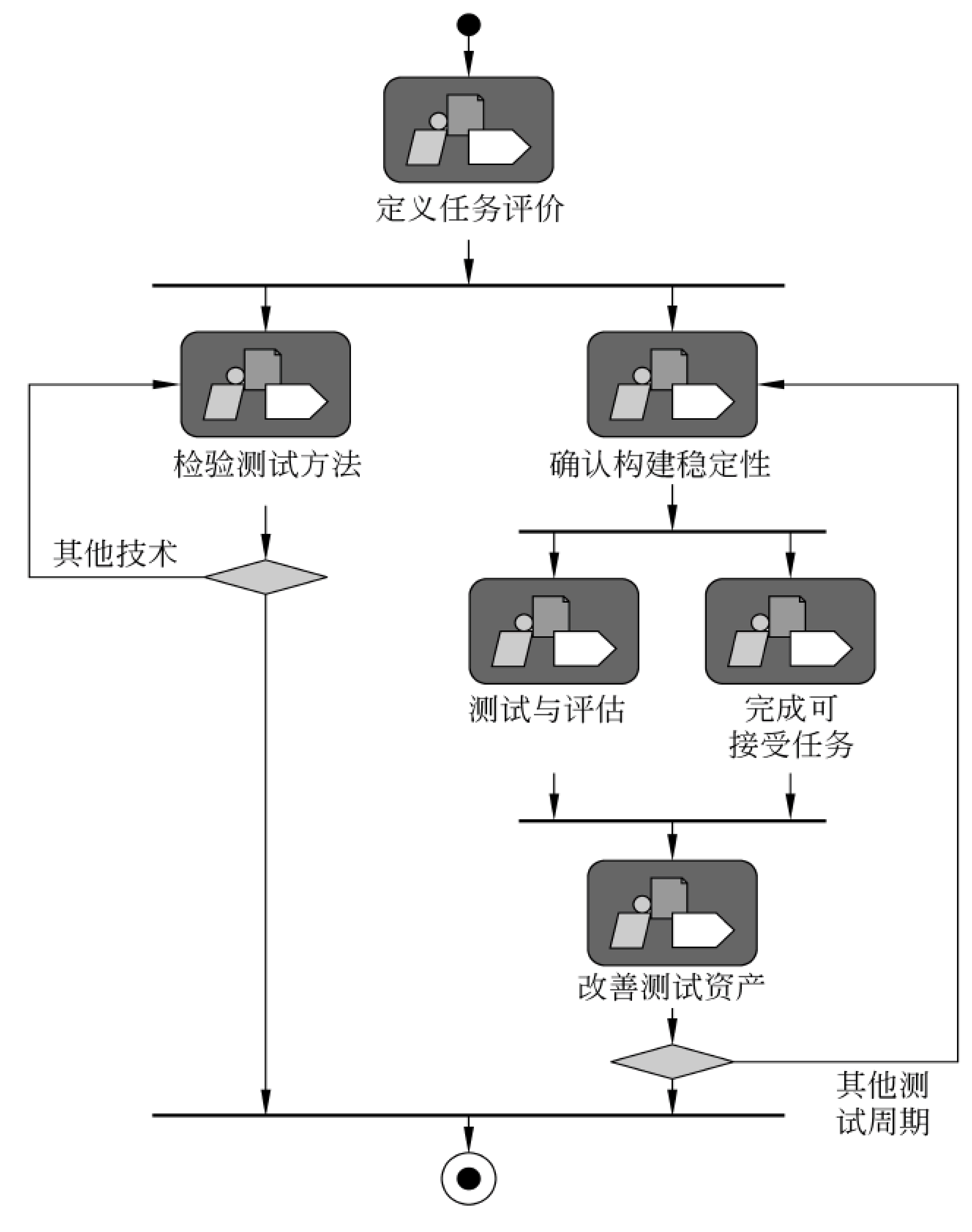


图 3-8　缺省测试规程

图 3-8 显示了用于 RUP 中的一次迭代的缺省测试规程的示例。此规程根据不同情况而有所不同。包含了许多步骤，说明了规程详细内容。对于测试规程，这些步骤是定义任务评价，检验测试方法，确认构建稳定性，测试和评估，完成可接受任务和改善测试资产。

对于每个项目，实现流程的方式意味着被选择的活动子集和工件，相关的角色以及谁来实现这些角色在开发用例和软件开发计划中规定。这也可以应用于测试流程。履行测试设计员角色职责的人员决定测试指南。测试经理负责指南集中的系统测试的执行，并管理不同的测试角色（谁在什么时间做什么）。其他角色还包括测试分析师和测试人员，在每个角色中，都要描述职责、相关技能和任务（包括可能的任务合并）。考虑到提前展开的任务的数量，每个项目的所有角色需要进行充分地分配，因此对于测试团队的成员，就可能共享角色和/或参与到多个项目中。

3.5.5 验收测试

验收测试是在软件部署之前的最后的测试。主要的目标是验证软件是否已准备好，可以被最终用户用于执行其设计的任务和功能。

验收测试被放在移交阶段，并且是部署流程的一个主要部分。RUP 定义验收测试不够充分，只是作为系统测试用例子集的重新运行。测试人员需要在一个类似产品的环境中执行这些用例。在验收测试期间，考虑工件产品的产品验收计划是很重要的。项目经理在项目的先启阶段开始编写产品验收计划。验收测试的相关主题是验收标准、接受的工件和评价方法。

RUP 没有提供用于在此阶段部署工具的指导。验收测试的实施可以部署在系统测试中所使用的相同工具。

3.5.6 复审

如 RUP 第五个最佳实践经验，RUP 将质量保证认为是在整个系统开发过程中要被执行的事情，质量保证计划是作为系统开发计划的一部分编写的，特别是控制过程的质量。复审在 RUP 中描述得很详细，定义了三个角色：复审协调员、管理复审员和技术复审员。复审协调员协调和管理复审过程，管理复审员主要检查项目计划和报告，技术复审员复审实际的系统开发产品（如业务用例模型、业务分析模型、需求、构架、设计和代码）。

3.6 RUP 测试解决方案

为实现高效的自动测试，必须有好的测试软件。测试软件可以买商业测试软件，也可以自己设计测试软件。无论哪一种测试软件，都要达到高于手工测试的效果，否则得不偿失。

目前市场上很多公司提供了完整的软件测试解决方案，著名的有 IBM 公司、Parasoft 公司等。这里简单介绍一下 IBM 公司推出的 Rational 测试解决方案，它提供了完整的系统级功能、性能测试和测试管理平台工具，其整个架构体系如图 3-9 所示。

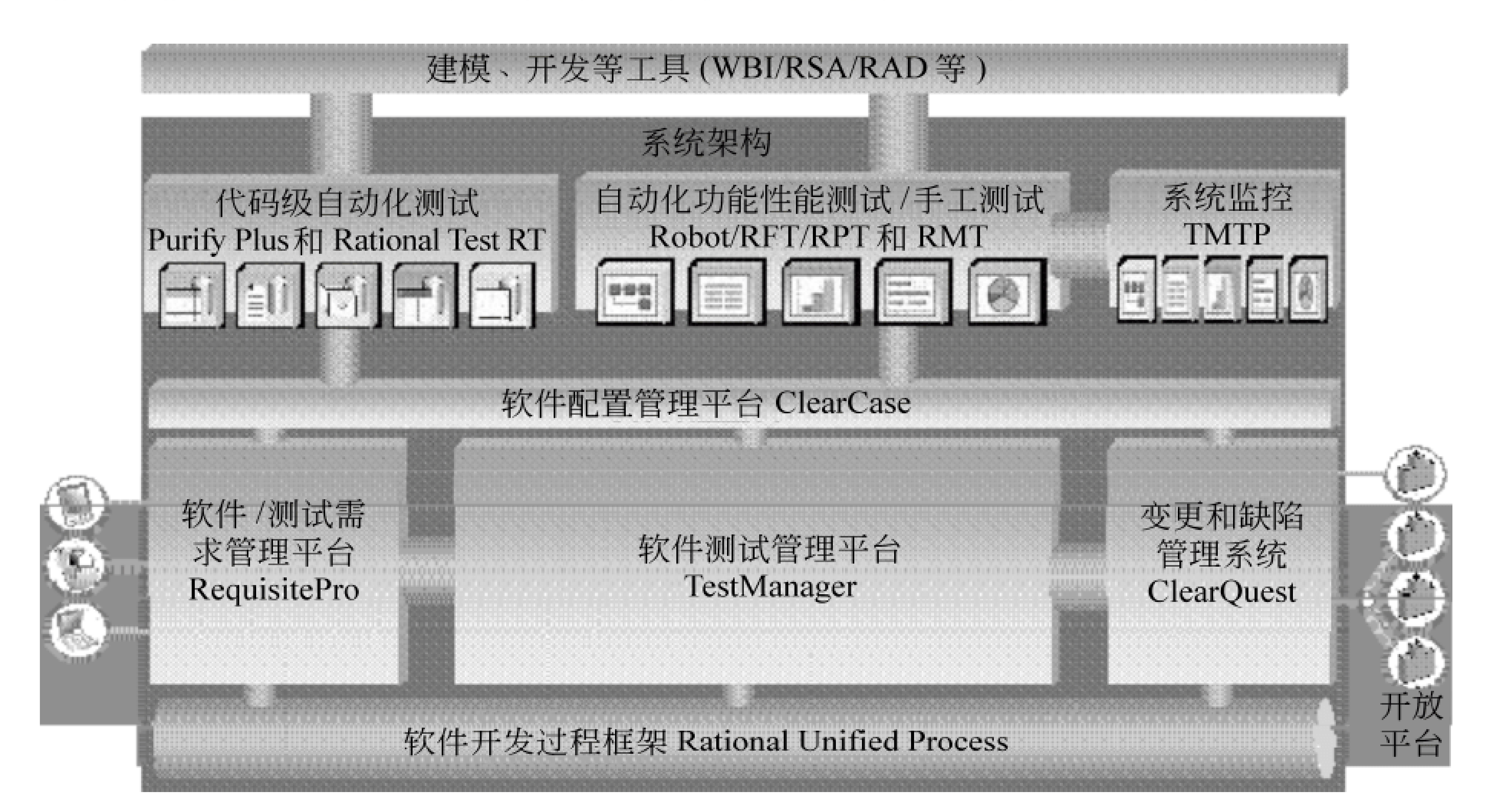


图 3-9　IBM Rational 软件测试体系架构

IBM Rational 软件测试解决方案以 RUP 的软件工程方法为基础。

首先，IBM Rational 软件测试解决方案提供了以 TestManager 和 ClearQuest 为核心的测试管理平台，完成从测试输入、测试计划、测试设计、测试执行到测试结果分析的整个测试流程的统一管理，提供对项目需求、变更请求、测试资料及其他数据的共享，提高了团队的工作效率。

其次，Robot 以自动化的功能和性能测试脚本录制、回放为基础，实现企业的自动化功能和性能测试，帮助企业解决回归测试和大批量的数据驱动软件测试所带来的工作量和工作效率问题，Tivoli Monitoring for Transaction Performance（TMTP）完成软件的性能检测和分析工作，进一步帮助系统测试人员分析、定位、解决系统性能问题。

第三，IBM 公司在 2005 年的 Atlantic 版本中新推出的功能测试工具 Rational Functional Test、性能测试工具 Rational Performance Test 和手工测试工具 Rational Manual Test，第一次为 Java 和 Web 测试人员，提供了与开发平台（Eclipse）无缝支持，具备完整的自动化功能、性能测试和手工测试的管理能力，而且其中还推出了 IBM 公司许多最新专利技术，例如基于 Wizard 的智能数据驱动的软件测试技术、提高测试脚本重用的 ScriptAssurance 技术等，同时它也提供了对开放软件测试架构 Hyades 的支持。

IBM Rational 的软件自动化测试解决方案致力追求测试工具和测试成功经验、测试流程的统一，上面阐述的每个测试成功经验和测试流程环节，都可以通过 Rational 的测试工具以及工具间的完美集成辅助完成。IBM Rational 的软件自动化测试工具如图 3-9 所示，其最大特点是通过一套完整的软件测试工具，在实现测试管理流程的基础上，同时涵盖了功能测试、性能测试和可靠性测试的自动化测试需求，通过工具之间的集成完成测试资源的整合，帮助测试团队运用 IBM Rational 的测试成功经验。

由此可见，IBM Rational 的软件开发平台通过实现了对整个软件开发全生命周期，包括从业务分析、需求管理、架构设计到系统构建、测试、部署的全方位支持，为企业提供了软件生产生命周期全部的质量保证。

3.7 RUP 使用技巧

在 RUP 中对测试流程的描述为：软件测试工作要通过制定测试计划、设计测试、实施测试、执行测试、评估测试几个阶段来完成。

1. 软件需求分析阶段

软件需求是开发工作和测试工作在制定计划、开展工作时所共同参照的源头和依据，只有在源头上控制好，才能保证后续工作的平稳开展。测试人员应从软件生命周期的需求阶段就开始介入，因为测试人员的计划测试工作通常在该周期后期开展。如果前期不参加，无法了解系统的真正需求，无法通晓整个系统的架构，造成测试困难。测试人员与需求撰写人员共同工作，在需求完成以后，审查以及理解需求。早期的审查以及建模可以暴露很多关于一致性、完整性和模糊性的 Bug，这个时候修补这些 Bug 付出的代价还十分小。测试人员在需求阶段的任务：全面了解系统需求，与客户进行一定沟通，从客户角度考虑软件测试需要达到的验证值。

参与软件需求调研，以测试角度分析需求的可测性；对不明确问题与客户或项目经理协调解决。一旦当前阶段测试工作的范围确定下来，就可以开始考虑测试需求的整理——也就是明确地定义现阶段要“测什么”。测试需求的确定将为我们制定一个可供衡量的标准，这些标准可衡量进度时间表、分配资源以及如何确定某个阶段测试工作是否可以完成。当然，还有更重要的一点就是已被确定的测试需求，它是进行测试用例设计和考虑测试覆盖的依据。整理测试需求的第一步，就是要“测试需求”。这里的“测试需求”中的“测试”是一个动词，指的是对软件需求本身的检查，这是测试工作的开始。

2. 软件分析设计阶段

测试人员除制定测试计划书等本职工作外，还有一个必不可少的任务，那就是将系统的可测性落实到书面文档，以备将来编写测试用例。之所以要这么做，是因为加强测试用例在测试过程中的地位、强化测试用例管理过程，如果编写测试用例直接参考需求规格说明书或分析流程图，这样跨度大，难度也大，是造成测试用例不完备、覆盖范围小的重要原因。

如果公司采用 IBM 公司的 Rose 等建模分析工具，在工具的帮助下，这个工作比较好执行；如果没有，那么测试人员更有必要编写一份《软件功能点测试描述书》。它是软件的详细测试分析文档，其主旨是将系统分析人员的分析文档加工站在测试角度的功能点分析文档，重要的是描述对系统分解后，每个功能点逐一地校验描述，包括何种方法测试、何种数据测试、测试结果期望值等。这些信息都是描述性的，无须具体数据用容易理解的自

然语言，明确地描述一项需要测试的内容。对于多项测试内容，应尽可能地剥离开来。它的作用是据此编写测试用例，以及测试执行时的参考依据，它直接来源于需求，覆盖最全，也最贴近客户。

软件开发阶段重点工作是编写测试用例。如何有效控制测试用例的流程？应该遵守的原则有以下几点：

首先，从覆盖率来说，测试用例库的用例要达到最大覆盖软件系统的功能点。编写测试用例时，基本就是将《软件功能点测试描述书》中的每个功能点进行操作上的细化：一是从步骤上描述到达校验点的方式，二是从内容上以何种数据校验功能点。

其次，从数量来讲，测试用例不能太少，必须能覆盖系统需求，以免测试人员不能在测试开展的整个时期按照用例执行。应该说测试用例的数量很难用数学模型来模拟，更没办法用具体的数字去衡量，只能掌握一个原则：系统需求必须覆盖。

再次，测试用例的完成并非是一劳永逸的，因为测试用例来源于测试需求，而测试需求的来源包括了软件需求、系统设计、详细设计，甚至包括了软件发布后，在软件产品生命周期结束前发现的所有软件缺陷。来源的多元化注定了测试需求是非常容易发生变化的，一旦测试需求发生变化，则测试用例必须重新维护。

最后，说一下测试用例格式上除一般说明外的几个要点：一是要制定适合本公司的测试用例模板，统一风格和延续传统；二是模板要有关键字索引，以方便按关键字分类查找，如测试方法（分手工/自动两种），使用的阶段（集成/系统测试阶段）；三是测试用例要有状态跟踪，如执行失败要链接到缺陷报告；四是测试用例的修改及运行都有日志记录。

3. 软件测试阶段

测试负责人划分不同的测试阶段（如集成测试、系统测试、回归测试、性能测试等），再划分不同的子测试周期（如前两个星期做冒烟测试，可手工，也可自动；接着做第一模块功能测试等），再为项目测试人员分配测试用例，测试人员则按照详细的用例文档去执行测试，这里要遵循的几个原则是：

① 健全严格的规章制度，来保证测试执行者严格按照测试用例执行测试。这并不妨碍测试者创造力的发挥，因为前期用例设计和编写就是项目全体测试人员智慧的结晶。众多测试工程师加班加点辛苦地工作，也主要是发生在这一阶段。如此实施，即便没有找出所有的缺陷，不能解决根本问题，也会大大提高测试执行效率。

② 如有对用例认识模糊或认为有遗漏的地方，必须经测试负责人或项目变更管理委员会同意方可更新用例库。

③ 测试负责人每日负责跟踪本测试子周期或阶段的测试用例执行情况，以及每日提交的缺陷报告，根据执行进展状态以及缺陷数量或严重等级与项目高层或其他人员交流，商议解决途径，并确定或调整未来的测试任务。

④ 测试执行者负责执行自己所分配的测试用例，还要负责跟踪这部分缺陷的修改情况，根据其状态不断验证软件功能点。

⑤ 由于不同公司开发产品的特殊性，也许需要特殊类型的测试，如安全测试，甚至代码级单元测试等，这些需要酌情考虑测试用例的编写，以及测试的执行。

4. 软件验收阶段

除了提交软件测试评估报告(各种类型测试结果的评估报告)这些传统工作外,此时还要集中时间更新测试用例,更新整个测试周期中一切需要更新的内容,以方便未来新版本的测试,实现软件测试用例的版本化控制。即便是软件提交客户后没有后续版本,也需要后期维护,维护阶段需要重新测试某些功能点,如果用例不准确,碰巧进行此项工作的又是个新员工,就会产生一系列问题。测试执行时遵循测试用例,执行完成后更新用例库,是测试部门的整体工作流程健全规范的主要表现。

3.8 小　　结

本文首先讲解了传统软件测试的问题,为了解决这些问题,RUP 提出了软件测试最佳成功经验,并由此介绍了 RUP 的测试理念,以及与质量保证的关系。在此基础上,提出了 RUP 软件测试流程和软件测试评测方法。

最后主要介绍了测试角色和四级测试,读者可按照角色定位学习测试知识。本教材架构按照四级测试来讲解 RUP 测试实践,符合读者的思维习惯。

RUP 认为高品质软件,需要完整的软件开发过程和整合的软件开发平台来共同铸就。本章介绍了 IBM Rational 软件开发平台,它是以国际标准和开放平台为基础,为软件产品的开发和生产过程提供了质量保证。IBM Rational 主要为软件测试团队提供测试成功经验、自动化测试工具和全方位的咨询服务三方面的支持,最终实现:一个测试团队,基于一套完整的软件测试流程,使用一套完整的自动化软件测试工具,完成全方位的软件质量验证,这正是 RUP 测试解决方案的精髓和终极目标。

习题与思考

1. 传统软件测试的一般过程是什么?
2. 传统软件测试有哪些问题?
3. 简述基于 RUP 的软件测试技术核心的三个最佳成功经验。
4. 简述 RUP 软件测试流程。
5. RUP 软件测试评测包括哪两种方法?其目的是什么?
6. 什么是 RUP 全过程质量保证思想?
7. 质量保证跟 RUP 有什么关系?
8. 如何有效控制测试用例的流程?
9. RUP 有哪四级测试?这些测试的目标是什么?
10. 叙述 IBM Rational 软件测试解决方案。

第4章 手工测试与自动化测试

自动化测试案例

微软在自动化测试方面是个良好的典范。例如开发 Exchange Server,常常要做一种压力测试,需要几万个甚至几十万个用户同时把 E-mail 发送到服务器,以保证服务器不会出现死机或崩溃的现象。可是,需要几万人同时发送 E-mail,这在现实生活中很难人为实现。但是,利用测试工具就可以非常容易地做到。测试工具可以自动产生几万个账号,并且让它们在同一时间从不同机器上(一个机器上可以有多个不同账号)同时发送 E-mail 信息。

IE 浏览器也是同样的情况,一般用户经常用到 IE 浏览 Web 页面,微软要求 50 000 个用户同时浏览一个 Web 页面,以保证网站服务器不会死机。一般来说,找到 50 000 个用户同时打开一个网页是不现实的,就算能够找到 50 000 个测试者,成本也非常高。但是通过测试工具则很容易做到,并且工具还可以自动判断浏览结果是否正确。

我们知道,基于 RUP 的软件测试三个成功经验,其中重要一条就是自动化测试。要求使用一套完整的自动化软件测试工具,来完成全方位的软件质量验证。自动化测试既包括技术方法方面,又包含管理方面;更重要的是,软件测试自动化是软件测试领域必经的发展阶段,随着应用软件程序规模的不断扩大,业务逻辑的不断复杂,以及从业者协作关系的日益重要,在软件的开发周期里适当使用自动化测试是非常必要的。

软件测试自动化已经成为国内软件工程领域一个重要课题。不言而喻,业内人士都意识到软件测试工作走向成熟化、标准化的必经之路就是要实施自动化测试。我们不可推翻测试思维在测试工作中的指导地位,而且自动化测试不可能完全代替手工测试,这两者在今后很长时间内都将作为互补型"伴侣"。本章简述了手工测试基础、自动化测试的基础理论和技术,自动化测试实践,阐述了企业引入自动化测试的条件,定义自动化测试过程的方法。读者应重点理解自动化测试基础和自动化测试技术,学习如何利用科学有效的 RUP 理论设计自动化测试。

4.1 手工测试基础

在过去的十多年里,可以看到支持需求管理、应用程序设计和开发、代码分析,以及单元、系统和部署测试等方面的软件有了很大的进步,这些软件已经帮助开发人员改进了软

件质量和加速了软件的交付。因此自动化测试工具新的发展使得交付高质量软件变得更容易了。然而从目前来看，软件测试大多数仍然是手工进行的。

一般来说手工测试步骤是这样的，测试人员使用 Excel 电子表格或 Word 表格（前者用得多），记录测试步骤、期望结果、在要求的时间段内通过/失败的状态。它们要么手工，要么使用导入 Excel 电子表格或 Word 表格的软件，来汇集测试结果，分析测试结果，然后生成产品报告。

4.1.1 手工测试的必要性

手工测试有其不可替代的地方，因为人具有很强的判断能力，而工具没有。手工测试不可替代的地方至少包括以下几点：

① 测试用例的设计。测试人员的经验和对错误的判断能力是工具不可替代的。

② 用户体验测试。人类的审美观和心理体验是工具不可模拟的。

③ 正确性的检查。人们对是非的判断、逻辑推理能力是工具所不具备的。

还有其他情况也适用于手工测试：

① 测试很少运行。对于很少运行的测试任务，例如一年只需要测试一次，对测试自动化则是一种浪费。

② 软件不稳定。如果在某段时间内软件的界面和功能更新频繁，那么修改相应的自动化测试点开销较大，适于手工测试。只有当软件达到相对的稳定，没有界面性严重错误和中断错误才适合开始自动化测试。

③ 涉及感观方面的测试。例如界面的美观、声音的体验、易用性的测试等，这类测试很容易通过人来验证，自动化测试反而难以执行。

④ 涉及物理交互的测试。自动化测试很难完成与物理设备的交互，例如刷卡的测试等。

4.1.2 手工测试工具概述

也许读者读到这里会产生疑问：手工测试就是手工测试，为什么还要手工测试工具，这不是成了自动化测试吗？

这里有两点要说明：

首先，手工测试工具不是替代手工测试，而是用来加速手工测试速度，提高手工测试的正确度。换句话说，它只是手工测试的辅助工具，如同一个工人拿个钳子或其他工具组装机器一样。

第二，手工测试存在着效率低等弊端，使用辅助工具可以更好地提高效率。例如进行手工测试的时候，经常要用到 Word 或 Excel 文件来记录测试数据或测试计划等。这些文档不是结构化的，不像 XML 那些文档，因此当测试越来越多的时候，这些文档经常会导致维护难度加大、时间延长，对测试造成不方便。由于这些文档有许多都包含了应用程序中常用操作的重复内容。当需要对测试进行更新时，测试人员不得不搜索所有测试文

档，把更新内容剪切并粘贴。此外还有很多问题，例如 Excel 表格或 Word 表格的固定宽度的格式以及需要在每页复制的标题都给打印报告带来了困难，很难按照测试步骤组织脚本，没有一种标准方式确认在测试脚本范围中所预期的结果，很难或基本不可能在不同的测试脚本中重用测试脚本等。

因此按照基于 RUP 的软件测试经验——自动化测试，IBM 公司开发了 Manual Tester，它是市场上唯一专门设计用来改进手工测试的工具，主要帮助测试人员简单直接管理手工测试脚本的创建和维护。

1. 复用及模块化的原则

只要是自动化，就不可避免涉及测试模块化和测试复用的概念，IBM 公司鼓励将可以复用的测试内容进行模块化，这样测试内容就可以在多个测试和多个测试人员之间共享。Manual Tester 通过提供管理测试内容的框架来简化模块化测试和测试复用的开发，这样测试人员就可以创建更小的模块化测试，且不用担心在进行测试的应用程序更改时，会有大量的测试脚本需要更新。通用的测试内容会放置在复用控制面板中，并且可以被多个测试成员创建的多个测试使用。这样的好处有两个：

① 测试维护变得简单。例如，当进行测试的应用程序发生变更时，拥有模块化的测试可以减少需要更新的测试数量。

② 测试工具通过提供测试调用的机制，体现了测试复用的概念。建立模块化测试编制常用的运行任务，然后重用那些测试，将确保测试维护时的一致性和准确性。

为了帮助读者掌握测试模块化的原则，请看图 4-1。

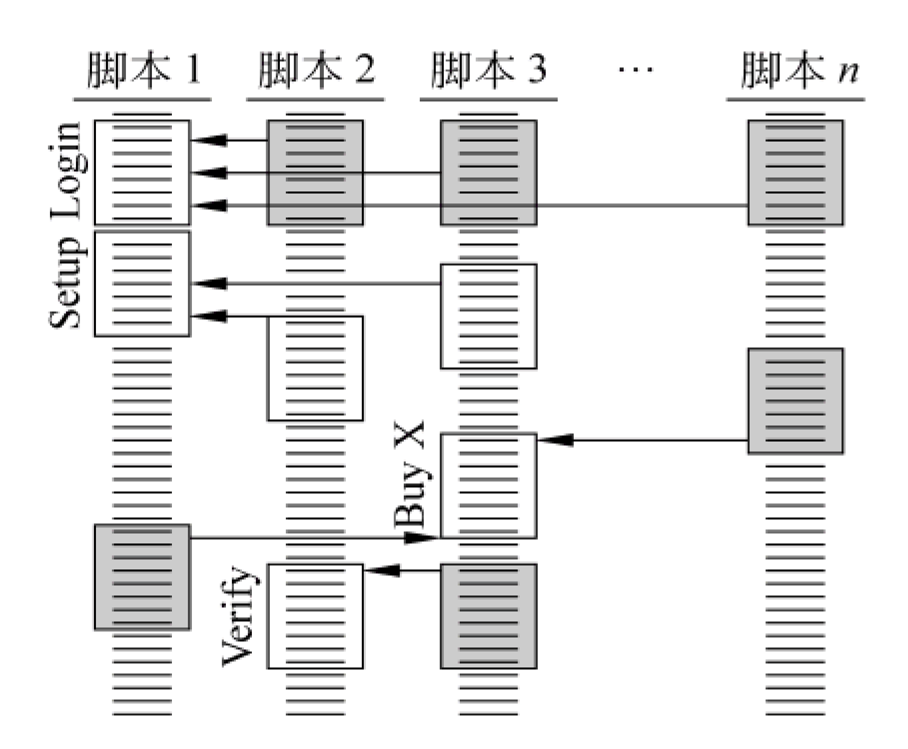


图 4-1 可重复流程作为可链接内容进行复用

图 4-1 中有 n 个脚本，每个脚本有一系列流程。这些流程有：

① 建立被测的应用程序流程，例如，启动和登录。

② 填写表单，例如，创建客户账户。

③ 导航到应用程序中的具体位置，例如，导航到订单输入屏。

④ 执行公共的验证，例如，数据库正确地表现事务吗？

⑤ 超过应用程序功能的期望值或界限，例如，当输入无效的信用卡号时，恰当的错误处理启动了吗？

⑥ 数据驱动测试脚本的一个子流程，例如，登录并为 50 个不同用户账户执行相同的事务。

⑦ 执行高阶的业务流，包括那些构成其他可重复流的内容，例如，下订单，卖股票。

可以看到，有很多流程是重复的，例如启动被测程序、程序登录等。IBM 公司定义为可重复流，并利用专业化的用户界面简化了模块化脚本的书写，这样让非程序员就可以轻易地以将重复流程作为链接内容而复用的方式书写测试脚本，如图 4-1 所示。测试人员

可以简单地拖曳(通过复用视图)或复制粘贴链接(通过 Ctrl+L 键)来创建模块化测试脚本。而且将可重复流程作为链接内容而复用是必要的,因为传统的人工测试和记录或回放自动化方法要求重复修改每个执行可重复流程的脚本。

2. 自动输入数据

手工测试中经常要对被测程序输入大量数据,这个时候最容易出现人为错误。Rational Manual Tester 提供了一个机制,专门辅助数据输入,可以减少数据输入错误。

在创建测试时,测试中必须输入的数据按照测试指导进行存储。在创建测试的过程中,需要输入的数据由向导直接从测试的应用程序中提取。换句话说,可以直接向测试中的步骤属性输入数据。在运行测试的过程中,在非手工输入数据的情况下,测试者可以从手工测试窗口中粘贴数据到测试中的应用程序里。在运行测试过程中,使用自动数据输入的好处是使测试更加快速准确。

3. 自动比较数据

手工测试经常要求测试人员向被测应用程序中输入数据,并验证结果信息与期望值匹配。对从进行测试的应用程序中输出数据的准确性进行核对,是手动测试应用程序中另一个容易出错的工作。如果测试人员不小心输入了错误的数据或漏看了不匹配数据,测试结果就无效了。IBM Rational Manual Tester 将数据输入和数据验证过程自动化,这样就减少了人为错误。

自动化的原理在于 Manual Tester 里面的内嵌数据比较器。该比较器可以用于比较预期的输出数据和被测程序输出的实际数据。同自动数据输入特性一样,预期输出数据存储在 Manual Tester 中,因此确保了验证的准确性。

4.1.3 手工测试工具的关键能力

由于 IBM Rational Manual Tester 是市场上唯一的手工测试辅助工具,它事实上为手工测试辅助工具制定了标准。它规定了手工测试工具的如下关键能力。

1. 支持测试验证组件化

RUP 的一个最佳经验是软件开发采用组件化的方法,而且也得到了大多数开发工具的支持。它允许不同的团队分别工作在不同的组件上,能够使开发人员更迅速地通过重用公共组件组装应用软件。一些自动化测试工具也使用这种方法进行测试开发。它们允许测试人员设计组件,然后他们就可以将这些组件组装起来创建一系列测试来验证整个软件应用程序。这种方法对于手工测试还不太适用,手工测试不是专门为组件而设计的。然而,使用 Manual Tester 便可以将组件的测试集合起来。这些组件记录了测试应用软件一小部分区域的步骤集。测试人员也可以重用每个组件,来组合成多种较大的验证一个应用软件用例的测试。

2. 支持自动单点更新

在多个测试之间共享的测试组件叫链接组件。当在测试中对一个应用软件的变化影响了一个用于多个测试中的链接组件时，测试员只需要更新测试组件中的步骤就可以了。Manual Tester 工具会自动将共享该链接组件的所有手工测试进行更新。通常，测试脚本维护，不论是手工的还是自动的，都是测试资源的最大消耗。这种单点更新能力，将会为业务分析师和测试团队带来极大的效率和较低的维护成本。

3. 强大的 Rich Text 编辑能力

Manual Tester 提供了强大的 Rich Text 编辑器以支持手工测试。Rich Text 编辑器也叫富文本编辑器(Rich Text Editor，RTE)，作为一个 Web 编辑器，它提供类似于 Word 的编辑功能，可以设置各种文本格式，因而受到用户的喜爱。测试员可以将文字变成粗体黑字，来指出要选择的按钮或菜单，也可以使用不同的字体指出要在测试执行期间输入的数据，并使用文件夹来分组步骤的逻辑块和嵌入式图像，这些图像提供了测试人员执行测试时的详细内容。强大的 Rich Text 编辑能力，使得手工测试更加方便。

4. 批量导入文档

如果已经在 Word 或 Excel 中记录了手工测试数据，使用 IBM Rational Manual Tester 可以批量导入这些文档，开始进行手工测试创建。

5. 方便数据输入

手工测试人员通常要向应用程序输入数据，来验证一个业务用例。这就会涉及大量的数据或数据字符串。使用 IBM Rational Manual Tester，测试人员可以通过 Manual Tester 的数据输入特性来提高手工测试执行的准确度和速度。简单地提供测试执行者可以自动重用的数据，来替代将数据手工输入到应用程序中。这将会加速数据入口过程，也确保了用于执行测试的数据被正确地输入。

6. 方便数据对比

除了输入数据之外，测试人员需要验证显示的数据或应用程序的输出是否正确。使用 IBM Rational Manual Tester，测试人员可以在记录测试数据的同时提供预期的输出数据，然后在执行期间使用 Manual Tester 提供的比较器，比较器是比较实际数据和预期数据的一种可视化工具，可以帮助测试人员确保数据输出与预先提供的数据是相同的。这些辅助数据用于验证测试是否被正确执行。

7. 支持分布式团队测试

随着软件规模的扩大和公司的国际化，大型软件开发团队分散在异地是常有的事情。测试团队必须利用所有的资源来保持竞争力，而不管这些资源是在哪里。IBM Rational

Manual Tester 可以帮助分散在各地的测试团队共享测试和结果，并且可以使用版本控制工具来管理测试的变更。

4.2　自动化测试基础

RUP 开发过程已经显示了比瀑布式开发的巨大好处，并已逐渐取代传统的瀑布式开发，成为目前最流行的软件开发过程。在迭代开发中强调在较短的时间间隔中产生多个可执行、可测试的软件版本，这就意味着测试人员也必须为每次迭代产成的软件系统进行测试。测试工作的周期被缩短了，测试的频率也增加了。在这种情况下，传统的手工测试已经远远不能满足软件开发的需求。如图 4-2 所示，当第一个可测试的版本产生后，测试人员开始对这个版本的系统进行测试，很快第二个版本在第一个版本的技术上产生了，测试人员需要在第二次测试时重复上次的测试工作，还要对新增加的功能进行测试，每经过一个迭代测试的工作量会逐步的累加。随着软件开发过程的进展，测试工作变得越来越繁重，如果使用手工测试的方法，将很难保证测试工作的进度和质量。在这种情况下应用良好的自动测试工具就势在必行。通过引入自动化测试，测试人员只要根据测试需求完成测试过程中的所需行为，自动化测试工具将自动生成测试脚本，通过对测试脚本的简单修改，便可用于以后相同功能的测试，而不必手工重复已经测试过的功能部分。

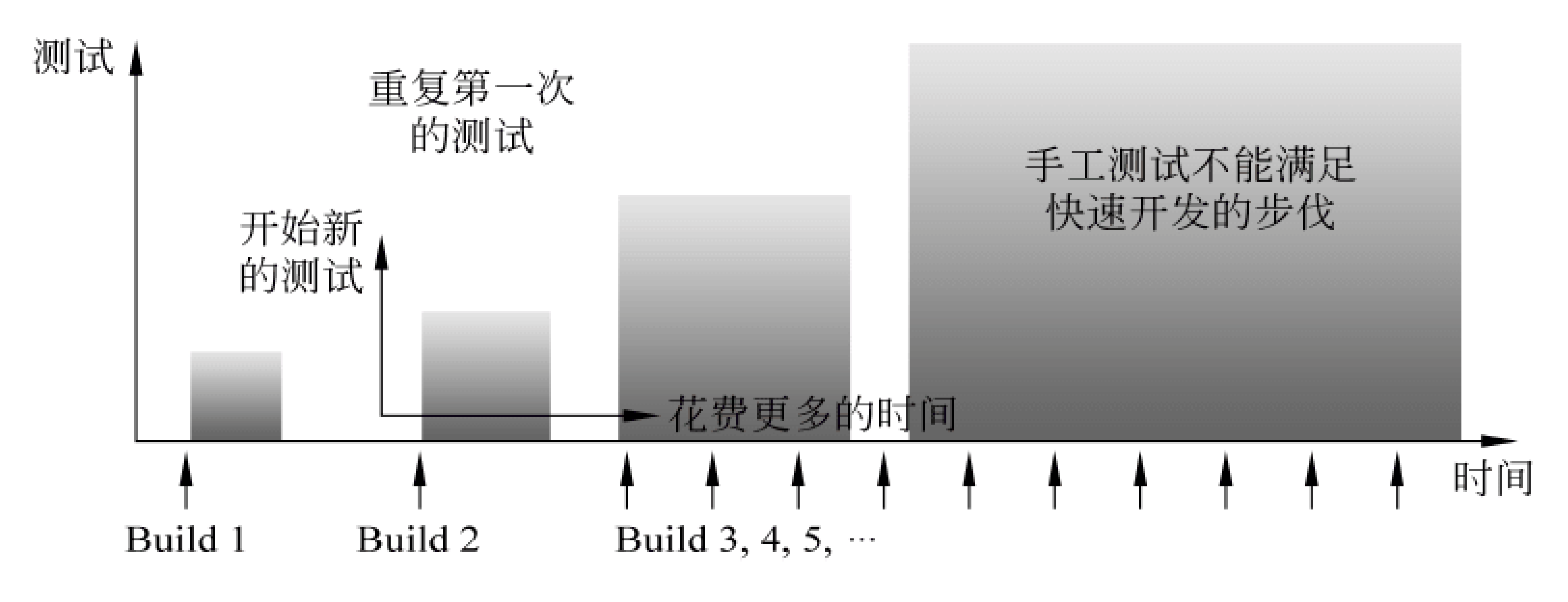


图 4-2　手工测试不能适应需要

同时现代的 GUI 开发技术已经非常先进了，它提供给开发人员快速开发的能力。这就意味着开发人员能够非常快速地改变应用程序，并将新的版本交给测试人员测试。实际上，很多公司每天都会有多个应用版本产生。如果还是使用传统的手工测试的方法，根本不可能符合软件快速开发的要求。

现代企业非常依赖非常复杂的计算机基础设施。如表 4-1 所示，一个典型的企业可能依靠多个应用程序，运行在不同的系统上，使用几种不同的前端客户端，涉及大量的业务过程，并且与很多种数据集交互。可能的组合高度复杂，需要成百上千的测试场景。

表 4-1　典型的企业应用测试

组　　件	数　　量	事　　例
平台	1	Intel
操作系统	5	Windows XP/ME/2000/NT4/98
前端客户端	4	Internet Explorer 6，Netscape 7.1,Java，Visual C++
业务过程	5	Login，Search，Order Entry，Order Confirmation，Order Fulfillment
数据集	15	Usernames,Passwords,Search Strings,Order Numbers,Ship Dates
需求的测试数量	1×5×4×5×15	1500 个可能的测试场景

当软件出现故障时，其代价是非常大的，包括销售额的下降、员工的低效率、客户的不满，以及开发、QA 人员的士气低落。在软件开发周期中，缺陷发现得越晚其代价越高。上线后发现的缺陷的改正成本可能比在设计阶段发现的高出 100 倍。自动化是提高软件测试过程的速度、精确度和灵活性的关键，使公司可以更早发现和改正缺陷。尤其在迭代式开发过程中，需要反复验证更改的系统或组件仍然保持应有的特性，引入自动化功能测试工具可提高测试覆盖率，缩短系统提交时间。美国质量学院 1995 年曾经对一项涉及 1750 个测试程序、700 个错误的测试进行了统计，结果表明采用自动化测试后测试工作总量减少 75%。因此引入自动化测试是很有必要，也是必然的。

4.2.1　自动化测试定义

根据软件质量工程协会关于自动化测试的定义，自动化测试就是利用策略、工具等，减少人工介入非技术性、重复性、冗长的测试活动。自动化测试的目标是对被测试系统进行自动测试，以达到较少的开销、较彻底的测试、提高产品质量的目的。根据以上定义，可以知道自动化测试就是借助于测试工具、测试规范，从而局部或全部代替人工进行测试，以提高测试效率的过程。

很多人一听到自动化测试就联想到基于 GUI 录制回放的自动化功能测试工具。实际上自动化测试技术的含义非常广泛，任何帮助流程的自动流转、替换手工的动作、解决重复性问题以及大批量产生内容，从而帮助测试人员进行测试工作的相关技术或工具的使用，都叫自动化测试技术。例如，一些测试管理工具能帮助测试人员自动地统计测试结果并产生测试报告，编写一些 SQL 语句插入大量数据到某个表中，编写脚本让版本编译自动进行，利用多线程技术模拟并发请求，利用工具自动记录和监视程序的行为以及产生的数据，利用工具自动执行界面上的鼠标单击和键盘输入等。

软件测试自动化的工作量非常大，而且也并不是适用于任何情况，同时软件测试自动化的设计并不比程序设计简单。自动化测试的本质是利用编码工作实现对已有代码的测试，因此，在工作前期还需要人为设定测试方案和确定测试用例，以达到利用自动化测试工具去自动测试程序的目的。自动化测试之所以称为自动化，是其利用计算机的优势，使重复执行的工作由程序代替人工去高效完成。

测试自动化可理解为测试过程自动化和测试结果分析自动化。

① 测试过程的自动化指的是不用手工逐个地对用例进行测试。

② 测试结果分析自动化指的是不用人工去分析测试过程中的中间结果或数据流。

4.2.2 适合自动执行的测试操作

测试自动化最适用的地方是对频繁的操作进行自动重复，免去了手工测试易错、费时、代价昂贵等诸多不便。软件测试自动化，即将软件测试中的一些测试操作交给测试程序自动执行。适合自动执行的测试操作有：

① 测试个案的生成，包括测试输入、标准输出、测试操作指令等。

② 测试的执行与控制，包括单机与网络多机分布运行，夜间及假日运行测试个案调用控制，测试对象、范围、版本控制等。

③ 测试结果与标准输出的对比。

④ 不吻合的测试结果的分析、记录、分类和通报。

⑤ 总测试状况的统计，报表的产生。

根据对上面操作的总结，RUP 推荐以下几个测试类型适合实行自动化测试：

① 重复性最大，例如数据的边界值测试、回归测试等；

② 冒烟测试，例如每个发布版本提交测试前的基本功能确认；

③ 配置测试，例如需要在不同支持平台的测试；

④ 复杂的测试，例如难以手工执行或者容易出错；

⑤ 需要对测试结果做电子记录的测试。

系统测试级别的回归测试是自动测试有效应用的一种情况。回归测试设法验证改进后的系统提供的功能是否仍然按照规定执行，系统在运行中没有出现任何非预期变化。自动测试几乎可以不加改动地重用先前的测试用例和测试脚本，以非常有效的方式执行回归测试。

选择适合实行自动化测试的情况

① 有些测试，虽然执行的时间不长，但过程烦琐，需要执行的动作非常多。例如，一个运行 10 分钟的测试，可能需要击键 150 次，打开 4～5 个窗口，切换操作。如果将其自动化，可以提高可靠性，也是值得的。

② 对软件进行的功能性测试，是测试软件系统在做什么。这些测试可以明确地知道应该在什么情况下输入什么，会有什么样的输出。这样的测试是非常容易实现自动化，也能从自动化中获得较大的收益。

③ 对软件进行的性能测试，包括在不同的系统负载下进行的测试。这些测试需要采用工具辅助完成，非常适合进行自动化。

④ 如果在测试中，运行 10％的测试需要花费 90％的时间，那么将这 10％的测试自动化是值得的。

4.2.3 RUP 自动化测试观点

企业购买了自动化测试工具，下一步就是要在公司内部推广自动化测试。那么，自动化工具能够给企业的测试流程带来多少改善呢？如何在测试工作中使用自动化测试工具

呢？RUP 提出了自动化测试以下关键要素：

① 自动化测试应该被看成一个软件开发项目，因为测试脚本是由代码编写出来的，而测试目的代码是自动化测试的根本任务。

② 有效开发并维护良好的测试脚本，是自动化测试的重中之重。

可以看到，自动化测试作为项目同样需要经历计划、设计、开发、维护、版本控制过程，具体而言，包括四个关键过程：

① 清晰的定义和可重用的实现过程；

② 获得企业组织管理上的支持；

③ 成熟的项目计划；

④ 稳定的结构设计。

一个定义良好并严格根据定义来实施的测试过程，是自动化测试成功的关键。在一个随机或非系统性的测试环境里，很难实施测试自动化；缺乏稳定的测试过程，或者拿起工具就开始录制脚本等，都是不正确的做法，这些行为将导致测试的失败。

RUP 提出的开发过程可以有效应用到测试和自动化测试中，因此根据 RUP 原则，得出优秀测试过程所应具备的几大要素：

① 初始计划——定义测试目标；

② 定义需求——确定测试什么，可落实到《测试需求说明书》；

③ 分析设计——决定如何测试，划分测试阶段、类型，以及测试方法等；

④ 实现——创建与实现测试，编写测试用例或开发测试脚本，并文档化；

⑤ 测试——调试测试（针对自动化测试脚本）；

⑥ 执行——执行测试；

⑦ 评估——评估测试结果并改进测试过程；

⑧ 配置与变更管理——测试脚本的版本控制和测试缺陷的跟踪；

⑨ 环境——定义支持测试所需的环境。

在实施自动化测试前，可根据上述内容定义软件项目的自动化测试过程，做到每个项目都有所规范，任何测试成员都据此实行。至于内容详细程度和文档格式，不必完全统一，重要的是内容符合规定的原则，并且在实际工作中贯彻执行。

4.2.4　自动化测试的标准

由于一个软件需要一个测试团队来完成测试，如果没有标准，这个测试人员编写脚本随意命名，并放在其计算机里，那个测试人员又用他的名字命名脚本，并放在服务器里。那么显而易见，他们编写的脚本就没法复用，他们之间就没法交换工作，甚至增大了脚本变更产生的维护工作量，不能跟踪测试资源的利用，造成实施上的混乱。因此有必要确定自动化测试项目的标准。

根据自动化测试的情况，自动化测试标准可包含两方面：

① 测试资源相关的标准；

② 测试过程相关的标准。

测试资源相关的标准，一般来说，包括表 4-2 中的内容。

表 4-2　测试资源相关标准

标　　准	重　要　性	要　　点
文件命名习惯	以唯一的标记来标志测试文件；方便版本控制	使用易于理解的词汇进行命名，反映测试，如测试用例、测试类型等；避免使用词汇名或变量信息等难于理解的词汇，如版本号或发布号等；命名规范可应用于单一的项目或整个项目
测试脚本编码规范	保证脚本文档的一致性，方便调试、复用和维护	一般采用和工具代码相似的标准来规范脚本代码；创建明确的标准来规范脚本文档，包括头文件的内容和脚本注释的风格等

文件命名习惯举例如图 4-3 所示。

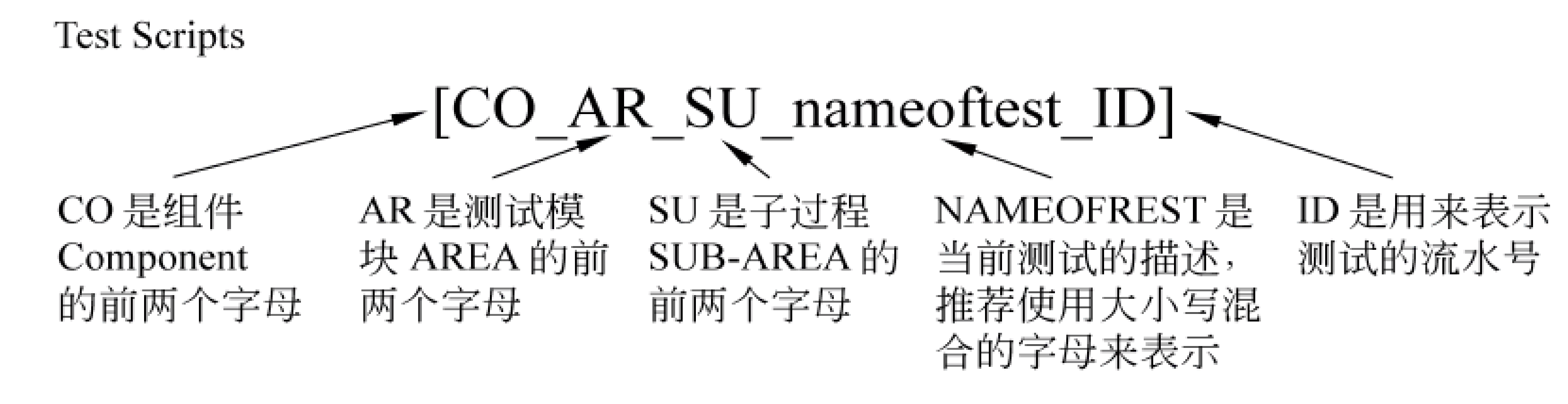


图 4-3　文件命名举例说明

例如：AD_IN_SY_synwithadm_01 表示一个 Administrator 组件、Integration 区域、Sychronizer 子区域，测试同步器在管理模块的用例，序号是 1。

AD_IN_SB_sybaserepo_02 表示一个 Administrator 组件、Integration 区域、Sybase 子区域，测试基于 Sybase 的组件库应用，序号是 2。

以 Rational Robot 为例说明测试脚本编码规范：Sub Name 是对一个子过程的描述，使用大小写混合的字符来命名即可。

例如：Declare sub writeFile (filename As String，data As Variant) 表示子程序名称为 writeFile，有两个参数，filename 是一个字符串类型的数据，data 是变量类型的数据。

标准脚本案例

```
'                  Test Automation Script
'                  ------------------------
'  SCRIPT:
'  AUTHOR:
'  DATE:
'
'  PURPOSE:
'
'  PARMAETERS:
'
'  PRE- AND POSTCONDITIONS:
'
'  MAINTENANCE:
'  ----------------------------
```

```
'   AMENDMENTS
'   <id><name><date><what&why>
'   ---------------------------------------------
```

Test Automation Script 标题告诉我们这是什么。

SCRIPT 关键字后面是脚本的名字。

AUTHOR 和 DATE 关键字后面是脚本作者的名字以及脚本的编写日期。明确标识出脚本作者的名字是有意义的，可以向作者提问或寻求帮助。

PURPOSE 关键字后面是对脚本目的的简短说明。通常说明不超过 1 行，如果说明较长，则有可能脚本的功能太多，最好分为两个或更小(可能复用性更好)的脚本。

PARAMETERS 关键字栏目中说明了脚本接收或返回的参数。参数说明有助于区别传入参数或传出(返回)参数。

PRE-AND POSTCONDITIONS 段说明脚本被调用时应具备的状态以及脚本完成后的状态。如果脚本用户希望脚本每次运行正确，这个信息特别重要。例如，脚本可能希望应用程序显示一个特定会话框，但在退出时显示不同的会话框。如果使用脚本的用户不知道这一点，有可能在要求的会话框没显示时就调用脚本，这将引起测试事例失败。

MAINTENANCE 段包含任何有助于维护脚本的信息，有时为避免一些问题(如工具或软件应用的某个功能不能正常运行)，则需要以特殊的方式实现脚本。

AMENDMENTS 段包含脚本第一次完成后的所有修改信息。

id 表示脚本所有变化或为修改而增加的行。

测试过程相关的标准如表 4-3 所示。

表 4-3　测试过程相关的标准

标　　准	重　要　性	要　　点
标准桌面环境	测试需要运行在不同环境下； 方便维护	最大可能地让测试环境接近于产品的实际运行环境； 使用操作系统的默认设置； 使用视频的标准输出
文件存储位置	需要团队成员都能访问测试文件	创建共享网络驱动器来存储测试资源； 网络驱动器对每个成员皆可访问
测试资源的版本控制	需要对测试资源的变更有所记录； 必要情况下需要访问先前版本的资源； 同一时刻只允许一个人修改文件	对于修改文件，需要流程上的规范
评估测试资源	保证规范被遵循； 对于测试用例和测试自动化的正确性有所评估	项目进度的评估，需要对评估点进行标记

对于测试过程，还可以用企业级自动化测试规范来描述。企业级自动化测试规范是对企业的测试流程及规范进行高标准的定义和描述，它定义了组织的测试目标、实施方式及遵循的标准，并包含了自动化测试在整个测试过程里的具体实施步骤。

RUP 并没有规定非要由何人来制订自动化测试规范，实际上可以是任何有自动化测试技能或经验的人，例如自动化测试的倡导者、测试主管等。制订完成后，需要得到企业

高层如首席技术官(CTO)的审批。

以下是某公司的自动化测试规范样例。

XYZ 公司自动化测试规范

(1) 介绍

该规范定义了 XYZ 公司的自动化测试过程，适用于公司所有的软件测试活动，对公司的软件测试活动的方法和步骤以及测试资源进行文档化。任何测试活动都要遵循该规范规定的标准和结构，但是对于特定项目的测试活动，可制订项目级的测试策略文档。

(2) 目标

XYZ 公司的软件测试目标是通过定制标准衡量软件系统的功能及其他非功能指标，以适应公司的商务运作，并以此衡量过程，作为评测软件发布的通道；个别测试项目还需参考项目的相关测试策略及计划文档。

(3) 方式

XYZ 公司测试标准以 RUP 为参考，并符合 RUP 规范。

(4) 组织

XYZ 公司采用手工测试和 Rational 自动测试工具结合的方式实施软件测试。

XYZ 公司采用有资格认证的人员确定测试方案，并通过技能培训保证相关人员在各自测试区域得到最大限度发挥。

(5) 步骤

……

4.3 测试自动化技术

软件自动化测试是一种测试技术，是通过使用自动化测试软件按照预先设定的机制，自动对被测系统执行测试的一种技术。下面就所用到的技术进行详细讲解。

4.3.1 自动化测试工具

自动化测试相对于手工测试而言，其主要进步在于自动化测试工具的引入。自动化测试工具可以进行测试设计、实现、执行和比较等工作。所以测试工具的选择和推广使用应该给予重视。

自动化测试工具按照执行的功能分类如下：

(1) 静态分析工具

静态分析工具用于分析设计模型、源代码或其他源程序中包含的信息，能够生成相关数据流、逻辑流或者质量指标等信息，包括复杂性、可维护性等。

静态分析主要集中在需求文档、设计文档以及程序结构上，可以进行类型分析、接口分析、输入输出规格说明分析等。常用的静态分析工具有：McCabe & Associates 公司开发的 McCabe Visual Quality ToolSet 分析工具、ViewLog 公司开发的 LogiScope 分析工具、Software Research 公司开发的 TestWork/Advisor 分析工具。

（2）测试数据生成工具

这些工具可获取测试活动中使用的数据，并通过转化、析取、变换或捕捉现有数据，使这些数据作为参考依据，自动为测试工具生成可靠的测试数据，减轻测试员在进行测试时生成大量数据的工作量，加快测试工作的进程。

测试数据生成工具主要应用在测试的前端，为测试过程准备大量的可用数据，为正常的测试流程和系统压力、性能测试提供有效的输入。目前典型的测试数据生成工具有：Bender& Associates 公司提供的功能测试数据生成工具 SoftTest、International Software Automation 公司提供的 Panorama C/C++ 测试数据生成工具、Parasoft 公司提供的 C/C++ 单元测试工具 Parasoft C++ Test 及 Java TM 类测试工具 Parasoft Jtest 等。

（3）测试评估工具

测试评估工具用于动态测试过程中对测试的内容及测试覆盖性进行评测，为测试的充分性提供依据。一般来说，测试评估工具都是在开展了一定时间的测试之后对整个测试情况进行一个整体上的评价，为随后的测试及软件的质量评估提供依据。常见的测试评估工具有：Bell 中心开发的 C 程序测试覆盖分析工具 ATAC、IBM 公司开发的 Rational PureCoverage、Software Research 公司开发的 TestWorks/Coverage、Parasoft 公司开发的 Parasofttca 测试覆盖率分析工具等。

（4）集成化测试系统

集成化测试系统将多种测试工具融为一体，是一种功能较强的测试工具。常见的有：德国 IDT 公司开发的针对 FORTRAN 程序进行测试的集成化工具 SADAT、Microsoft 公司开发的对 Windows 应用程序进行自动测试的集成化测试系统 Microsoft Test for Windows、美国 Parasoft 公司开发的自动故障检测系统 Parasoft Insure++ 等。

（5）测试管理工具

测试管理工具能够用于辅助测试活动或工作的计划、设计、实施、执行、评估和管理。目前，比较有代表性的测试管理工具主要有：Microsoft 公司的 RAIDS、IBM 公司的 Test Studio、Mercury Interactive 公司的 Test Director、Silicon Valley Networks 公司的 Test Expert 等。其中，RAIDS 是专注于缺陷管理的工具，它提供了缺陷填报、缺陷查询等功能。而 Test Studio 是专注于对测试用例的管理的工具，它按层次组织测试用例，易于查询和运行测试用例。Test Director 和 Test Expert 是另两个测试管理工具，也提供了较好的缺陷管理和测试用例管理功能。

在开展软件项目测试时，熟悉各种成熟测试工具的性能，把握各种工具的应用与优缺点，就能以较快的进度完成测试项目，而在此基础上进一步开发适合自己项目的二次开发程序，更能缩短软件项目的测试周期，提高软件的产品质量。

软件自动化测试工具技术主要有：直接对代码进行静态和动态分析、测试过程的捕获和回放、测试脚本技术、虚拟用户技术和测试管理技术等。其中捕获回放技术、脚本技术、自动比较技术是构成现代测试工具技术的三大基石。

4.3.2 代码分析技术及插装技术

代码分析类似于高级编译系统，针对不同的高级语言去构造语言分析工具，在工具中定义类、对象、函数、变量等的定义规则和语法规则；在分析时对代码进行语法扫描，找出不符合编码规范的地方；根据某种质量模型评价代码质量，生成系统的调用关系图等。

代码分析最大的用途就是用于插装测试技术。因为插装测试为了确定插装的位置，需要对程序源代码进行分解和索引，从而可以标识每一个插装位置。采用词法，语法分析器进行代码分析和信息收集，信息存储借助数据库。IBM 公司的 PurifyPlus 工具有一项专利——目标代码插入技术（Object Code Insertion，OCI）就是应用了插装技术，它在程序的目标代码中插入了特殊的指令用来检查内存的状态和使用情况。这样做的好处是不需要修改源代码，只需要重新编译就可以对程序进行分析。这个专利技术解决了插装内容过多和性能下降的问题。因为在原有程序中插入大量非相关任务的代码，程序性能会有所下降。为了解决这一矛盾，OCI 技术通过插装功能单一化减少插装内容，并且对插装代码的性能进行严格要求和测试，降低对程序性能的影响程度。

代码分析同时还对插装内容进行设计，根据不同的需求进行不同的插装，即每次只能针对需要收集的信息进行相应功能段的插入，而不提供一次性插装，可同时获取多种信息。使得对于插装前后的代码量的控制可以在允许范围内。

插装测试类型有：

① 函数性能分析插装：在每个函数入口和退出处插入跟踪代码，维护一个计数器，统计函数调用次数；记录本次进入、退出时间，计算本次执行时间；维护该函数调用的 max、min、avg 等时间，该函数执行时间在总程序中所占比率。

② 函数调用序列跟踪插装：在每个函数入口和退出处插入跟踪代码，维护一个被调用深度等级，记录函数调用层次。

③ 函数控制流分支跟踪插装：在每个函数入口和退出、每个分支部分插入跟踪代码。

PurifyPlus 工具进行代码分析时，从插装内容定位到程序高层行为，可自动显示出被测程序运行时的高层信息，这样对程序的调试和可视化观察更有价值。其生成的语法分析器自下而上进行语法归约，当归约到不同的语法规则时，设计语义动作进行程序信息（类、函数、程序块等）提取，并存储入库。从数据库中获取程序基本信息后，主要进行项目全局类图的分析和局部类图的分析，并把分析结果以 UML 图的形式返回给用户。这里主要介绍全局类图的成图情况。

由于C++ 不具备强制单根继承结构，导致如果仅对继承而言，全局类图很可能已经无法用一棵树型结构代表，也就是说，可能存在很多棵树构成的森林，再考虑到加入包含的关系，整个类图的结构会变得更加复杂，因此必须寻找新的结构来完备地表示软件项目的类图结构。这里提出了图的一种新的存储方式：双向邻接链表法，如图 4-4 所示。

在显示全局类图之前，首先确定每个类的继承层次，层次深的优先访问，优先确定位置坐标。设计算法 dSearch，确定类的层次，同时深度优先遍历已建立好的类图结构

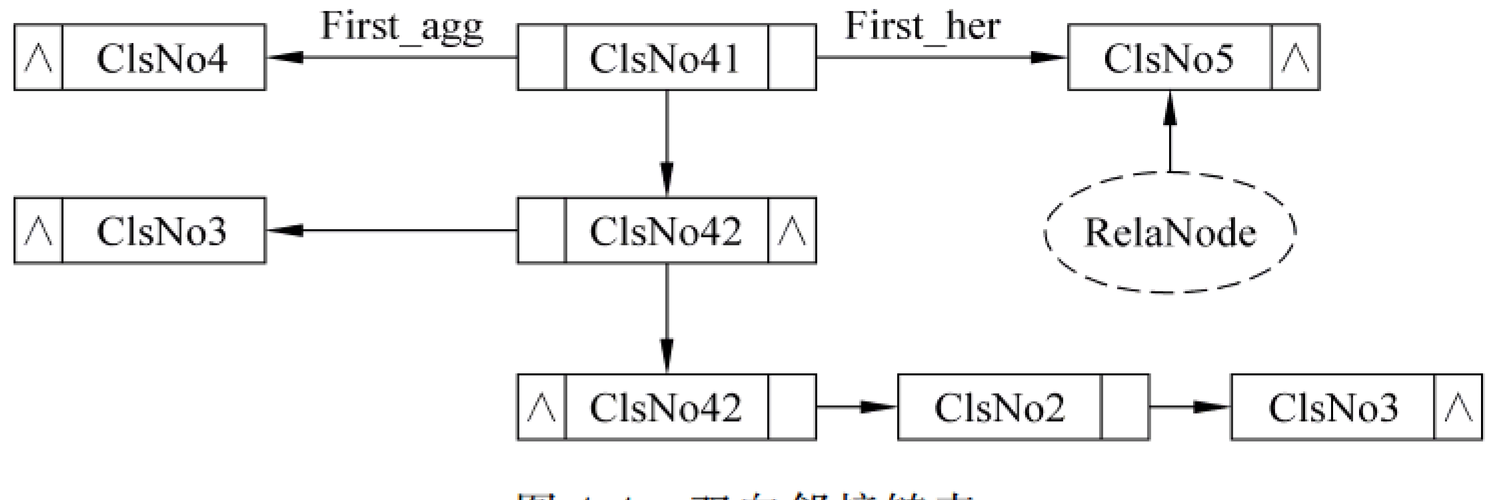


图 4-4　双向邻接链表

pGloList,确定每个类的继承优先级及属性优先级,为了保证全局类图的完整性,每个类节点设置访问标志。对于非连通类图来说,通过访问标志可以保证类图的每一个分支都得到完整显示。

4.3.3　什么叫脚本

在前面反复使用了一个名词,就是脚本(Script)。脚本原意是指表演戏曲、话剧、拍摄电影、电视剧所依据的本子,里面记载台词、故事情节等。这里是指使用一种特定的描述性语言,依据一定的格式编写的可执行文件,脚本也可被称作宏或批处理文件,它通常可以由应用程序临时调用并执行。例如 Office 的宏命令也是脚本。很多动态语言如 ASP、PHP、CGI、JSP 等都是脚本语言,因此脚本用在网页设计上最多,因为脚本不仅可以减小网页的规模和提高网页浏览速度,而且可以丰富网页的表现,如动画、声音等。举个最常见的例子,当点击网页上的 E-mail 链接时能自动调用 Outlook Express 或 Foxmail 这类软件,就是通过脚本功能来实现的。金山词霸网站的单词词典内容旁会有一个喇叭符号,点击它就可以听到英文诵读,这也是脚本在起作用。

需要注意的是,脚本其实就是一系列命令和指令,也是计算机程序的一种形式,但跟平时见到的一般程序像 Word 程序等是有区别的。它是动态程序的实现方式之一,动态程序一般有两种实现方式,一是二进制方式,一是脚本方式。

① 二进制方式是先将编写的程序进行编译,变成机器可识别的指令代码(如.exe 文件),然后再执行。这种编译好的程序只能执行、使用,却看不到它的程序内容。

② 脚本简单地说就是一条条的文字命令,这些文字命令是可以看到的(如可以用记事本打开查看、编辑),脚本程序在执行时,是由系统的一个解释器,将其一条一条地翻译成机器可识别的指令,并按程序顺序执行。因为脚本在执行时多了一道翻译的过程,所以它比二进制程序执行效率低一些。

脚本同平时使用的 VB、C 语言的区别主要有以下几点:

① 脚本语法比较简单,比较容易掌握;

② 脚本与应用程序密切相关,它会调用应用程序自身的功能;

③ 脚本一般不具备通用性,所能处理的问题范围有限。

测试脚本是一组测试工具执行的指令集合,由测试工具执行自动化操作而完成测试任务。脚本可以通过录制测试的操作产生,然后再做修改,这样可以减少脚本编程的工作

量。当然，也可以直接用脚本语言手工编写脚本。

4.3.4 录制/回放技术

代码分析是一种白盒测试的自动化方法，录制/回放则是一种黑盒测试的自动化方法。最初的自动化测试工具只提供了简单的录制/回放功能，目前的自动化测试解决方案几乎都是采用“录制/回放”的技术，并作为基本功能。

录制，就是记录对软件的操作过程，回放，就是重放录制的软件操作。启动测试自动化工具，打开录制功能，依照测试用例中的描述一步一步地操作被测软件，测试自动化工具会以脚本语言的形式记录下操作的全过程。依照此方法，可以将所有的测试用例进行录制。在需要重新执行测试用例时，回放录制的脚本，测试自动化工具依照脚本中的内容，操作被测软件。除了速度非常快之外，通过测试自动化工具执行测试用例与人工执行测试用例的效果是完全一样的。

录制需要将用户每一步操作都记录下来。这种记录的方式有两种：记录程序用户界面的像素坐标或记录程序显示对象（窗口、按钮、滚动条等）的位置，并且记录相对应的操作、状态变化或属性变化。所有的记录转换为由一种脚本语言所描述的过程，用以模拟用户的操作。回放时，将脚本语言所描述的过程转换为屏幕上的操作，然后将被测系统的输出记录下来与预先给定的结果进行比较。这种方法可以大大减轻黑盒测试的工作量，在迭代开发的过程中，能够极大地减轻回归测试的工作量。

录制只是实现了测试输入的自动化。一个完整的测试用例由输入和预期输出共同组成。所以，光是录制回放还不是真正的测试自动化。测试自动化工具中有一个检验功能，通过检验功能，在测试脚本中设置检验点，使得测试自动化工具能够对操作结果的正确性进行检验，这样，就实现了完整的测试用例执行自动化。软件界面上的一切界面元素，都可以作为检验点来对其进行检验。

捕获回放案例

程序中各模块都要使用的公共程序部分（如用户登录界面），用 Rational Robot 录制单击按钮的一行脚本：PushButton Click，“Type = PushButton; Name = Yes; VisualText = Yes”。在测试脚本中，可能会用到上千条这样的语句，测试工具通过按钮名字和显示文本来识别这个按钮。但当其中任何一个按钮或程序发生改变时，所有相关的脚本都会受到影响，它的改动会引起大量测试工作的返工，造成测试脚本的日常维护工作量急剧增大。

由于录制/回放技术使用的简单性，因此很多自动化测试工具厂商都宣传他们的工具非常容易使用，没有技术背景的测试人员只要简单录制测试的操作过程，然后播放录制好的测试脚本，就可以轻松地进行自动化所有的测试。

录制/回放为自动化测试提供了一个很好的途径，但由于其固有的缺点，使得它在自动化测试中的作用受到制约。现在来分析一下自动化测试不能单单只依靠录制/回放来完成的原因：

首先，是成本问题。通过录制建立的脚本，基本上都是用脚本语言以硬编码的方式编写的，当应用程序变动时，这些编码也随之需要更改，以至于测试人员要不停地修改测试脚本，这样一来问题就多了。随着软件规模的扩大，捕获/回放工具会产生大量的脚本，这么多的脚本维护起来，工作量将会变得异常庞大。因此维护这些录制好的脚本，成本是非常高的，高得几乎不能接受。

其次，所有的测试脚本都必须是在应用程序可以正确执行时才能录制，如果在录制过程中发现缺陷，测试人员必须向缺陷管理机制报告，等到该缺陷修正了，整个录制脚本的动作才能继续下去。在这样的情况下，如果仅依靠录制脚本来进行测试，效率低下。此外用到的测试用例，除了测试应用程序的图形用户界面，实际上没有任何用处。

最后，这些录制好的脚本并不是完全可靠，就是在应用程序完全没有变动的情况下直接播放，也可能因为一些意外状况而无法执行。如果录制脚本时，测试人员进行了错误的操作，则脚本就必须重新录制。

此外，通过录制的方式来建立自动化测试脚本的方式看似容易，但实际上会遇到下列问题：

① 测试人员大多不具备技术背景，难以完全掌握测试工具；

② 应用程序必须达到一定的稳定性，才能开始录制测试脚本；

③ 录制的测试脚本与测试数据的相互依赖性很强。

因此，这些录制/回放测试方法虽然最容易应用，但仅依靠录制/回放来完成自动化测试是远远不够的，于是又出现了功能和灵活性更强的测试脚本技术。

4.3.5 数据驱动技术及关键字驱动技术

如图 4-5 所示，录制回放的特点是录制下来的脚本包含了测试数据，数据驱动技术正是为了解决录制回放问题而提出的测试方法，它是目前自动化测试应用最多的架构模式，特点是把测试脚本和测试数据进行了有效的分离。

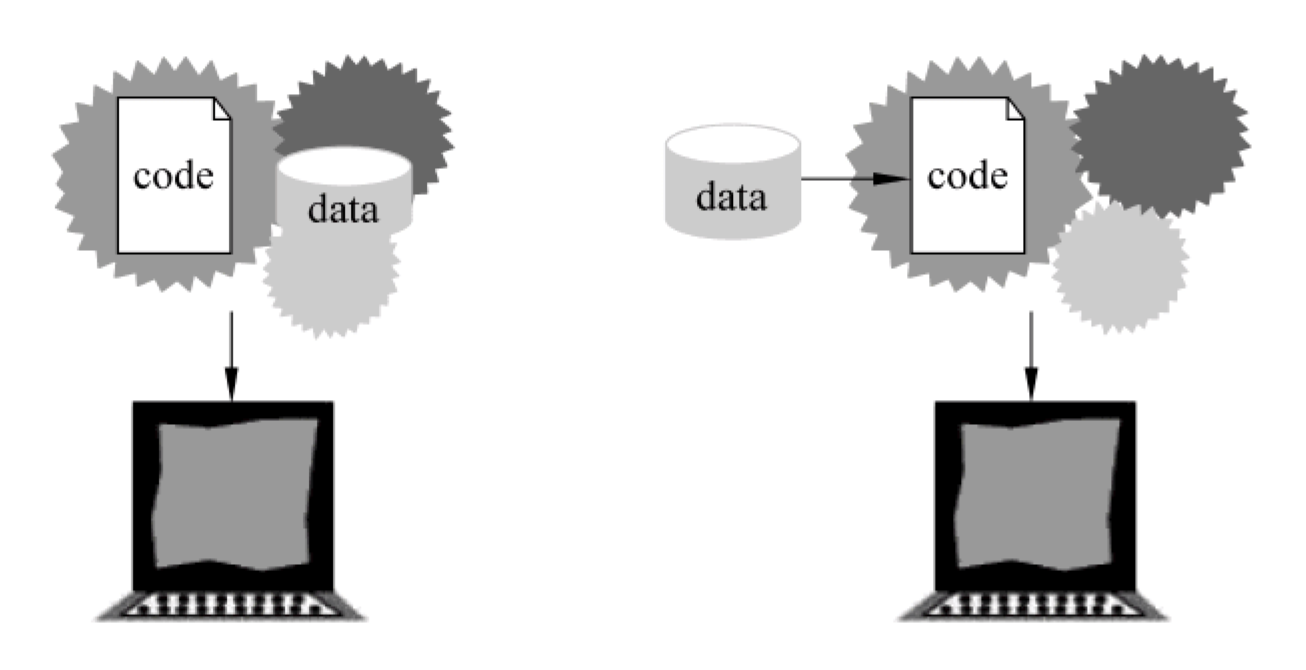


图 4-5 录制回放和数据驱动

数据驱动的工作过程为：数据驱动测试工具从某个数据文件（如 ODBC 源文件、Excel 文件、Csv 文件、ADO 对象文件等）中读取输入，然后输出测试数据。这就要求事先准备好测试数据，一般可以参考程序流程图或用例图来设计。流程图可提供测试系统需

要的数据信息，或提供数据间的依赖关系，根据关系定义需要的数据，例如数据库数据、输入框数据、期望数据等。然后确定这些数据是否合适，或从中摘录合适的数据子集。最后通过变量传入事先录制好的或手工编写的测试脚本中。其中，这些变量被用作传递（输入输出）数据，以验证应用程序的测试数据。在这个过程中，数据文件的读取、测试状态和所有测试信息都被编写进测试脚本里。测试数据只存放在数据文件中，而不是脚本里，测试脚本只是一个“驱动”，或者说是一个传送数据的机制。

数据驱动测试案例

问题：美国一公司在测试录制过程中，使用员工的唯一社会保险号为一个新员工创建个人档案。每次测试运行时，都会提示数据库中已经存在了相同社会保险号的记录。

解决方法：使用数据驱动的测试来向应用程序提供不同的员工数据，包括社会保险号。

问题：在录制测试时，删除了一条记录，在测试运行时，自动化测试工具将视图删除相同的记录，系统会提示“记录无法找到”的错误信息。

解决方法：可以在测试回放中，使用数据驱动的测试来引用不同于在录制时删除的记录。

关键字驱动（Keyword Driven）有时也叫表驱动（Table Driven），是数据驱动技术的变种，它是对数据驱动技术的有效改进和补充。关键字指存放在数据表的描述性文字，可以决定被用来测试的数据，这类数据被解释执行，而不是仅为测试提供参数。关键字脚本技术是将测试步骤细化、分解，在此基础上建立一个能够被大量引用的功能词句库。这样需要将数据文件变成对测试用例的描述，用一系列关键字指定要执行的任务。在关键字驱动技术中，假设测试者具有某些被测系统的知识，则不必告诉测试者如何进行详细的动作，只是说明测试用例做什么，而不是如何做。这样在脚本中使用的是说明性方法和描述性方法，描述性方法将被测软件的知识建立在测试自动化环境中。关键字驱动脚本的数量不随测试用例的数量变化，而仅随软件规模的变大而增加。这样使通常要编写大量测试代码才能完成的测试脚本，在采用关键字驱动脚本技术后，变成了几个词句的逻辑组合，使开发脚本变得简单，并且易阅读和维护。

由于关键字驱动技术是在数据驱动基础上发展起来的，吸取了数据驱动中将可变部分和不可变部分分离，以降低维护工作量的思想，将测试逻辑同测试脚本也分离开来。关键字驱动真正实现了数据与脚本分离，测试逻辑与测试脚本分离，这种分离使得分工更明确，并且避免了它们相互之间的影响，实现了测试的完全定制。使用模块化的测试脚本组织测试。

关键字驱动的自动化测试技术是一种截然不同的思想，这种模型的开发和实现与传统的测试流程相比可能是困难的，最耗时的，它需要将具体测试和自动化工具以及应用程序本身的变化完全隔离开来。但是这样的投资是一次性的，一旦开发结束并投入使用，它将带来的效益是巨大的，是自动化测试框架中最容易维护和使用的，而且可以反复运用于各种应用中，长期发挥作用。此外，现在已经有一些符合需求的商业化产品可供使用，减少了实现这种框架的困难。

4.3.6 脚本预处理

预处理是指一种或多种预编译功能,包括美化器、静态分析和一般替换。脚本的预处理是指脚本在被工具执行前必须进行编译。预处理功能通常需要工具支持,在脚本执行前自动处理。

美化器是一种对脚本格式进行检查的工具,必要时将脚本转换成符合编程规范的要求。可以让脚本编写者更专注于技术性的工作。

静态分析对脚本或表格执行更重要的检查功能,检查脚本中出现的和可能出现的缺陷。测试工具通常可以发现一些如拼写错误或不完整指令等脚本缺陷,这些功能非常有效。静态分析可以检查所有的缺陷和不当之处。类似于 C 程序设计中的 Turbo C 的语法检查功能。

一般替换也就是宏替换。可以让脚本更明确,易于维护。使用替换时应注意不要执行不必要的替换。在进行调试时,应该注意缺陷可能是存在被替换的部分中,而不是原来的脚本中。

4.3.7 自动比较技术

自动比较是检验软件是否产生了正确输出的过程,是通过在测试的实际输出与预期输出(如当软件正确执行时的输出)之间完成一次或多次比较来实现的,它是评价大量输出的唯一可重复而有效的方法。这个工作几乎不能靠人工来完成,即使是具备奉献精神的测试人员,在浏览了大量页面的输出、跟踪或转储后,也会变得疲惫不堪。进行测试自动化工作时,自动比较就成为一个必须的环节。

比较可以是简单的比较,仅匹配实际输出与预期输出是否完全相同,这是自动比较的基础。智能比较是允许用已知的差异来比较实际输出和预期输出。例如,要求比较包含日期信息的输出报表的内容。如果使用简单比较,显然是不行的,因为每次生成报表的日期信息肯定是不同的。这时就需要智能比较,忽略日期的差别,比较其他内容,甚至还可以忽略日期的具体内容,比较日期的格式,要求日期按特定格式输出。智能比较需要使用到较为复杂的比较手段,包括正则表达式的搜索技术等。现在的比较器功能较齐全,可以标识有差异的内容。但比较器并不能告诉用户测试是否通过或失败,需要用户自行判断。

自动比较的内容可以是多方面的,包括:基于磁盘输出的比较,如对数据文件的比较;基于界面输出的比较,如对显示位图的比较;基于多媒体输出的比较,如对声音的比较;及其他输出内容的比较。

有计划地进行比较相对随意的比较一般具有更高的效率和发现问题的能力。

4.3.8 测试自动化成熟度

许多公司在开展自动化测试时花费了大量的人力和物力,最终却发现收获颇微,其主

要原因在于自动化测试设计本身不合理。而当前流行的 CMMI(Capability Maturity Model Integration,能力成熟度模型集成)等通用模型,没有针对自动化测试领域进行详细阐述,自动化测试没有等级化的成熟度考量,缺少改进的指导与动力。

测试自动化成熟度,补充了这方面的不足,对测试专家/软件质量专家要进行测试过程自身评估和改进提供了极大帮助。对于刚进入测试领域的新人,也不失为了解测试规范要求、理清自身学习和发展思路的好参考。按照自动化测试的成熟度模型,自动化测试可被划分为 5 个级别:

1. 级别 1:录制和回放

这是使用自动化测试的最低的级别。

优点:自动化的测试脚本能够被自动地生成,而不需要有任何的编程知识。

缺点:会产生大量的测试脚本,同时当需求和应用发生变化时,相应的测试脚本也必须重新录制。

用法:当测试的系统不会发生变化时,可进行小规模的自动化。

2. 级别 2:录制、编辑和回放

在这个级别中,测试人员使用自动化测试工具来捕获想要测试的功能。将测试脚本中的任何测试数据,如名字、账号等,从测试脚本的代码中完全删除,并将他们转换成为变量。

优点:测试脚本开始变得更加完善和灵活,并且可以大大减少脚本的数量和维护的工作。

缺点:需要一定的编程知识。频繁的变化可能会引起"意大利面条式的代码",并且变更和维护几乎是不可能的。

用法:当进行回归测试时,被测试的应用有很小的变化,例如仅是针对计算的代码变化,应用程序界面没有发生变化。测试人员可以使用这种技术来快速编制一些测试脚本以检验脑子里的想法来探索预定的测试设计。通常如果适当的软件配置管理(CSM)与良好的内部构建设计相结合时,使用级别 2 的技术已经足够了。

意大利面条式代码的故事

意大利面条式代码的说法来源于汇编语言。汇编语言的一个主要问题是需要好几条汇编语句才能实现一条高级语言的语句。很多时候,汇编语言程序员发现通过将程序跳转到一些程序结构的中间,可以节约几个字节或几个周期。在确定这点(并对代码作了相应的修改)后,代码最后会包含一串跳转指令。如果在每条跳转指令和相应的目标地址间连上一条线,最后代码看上去会像一碗堆起的意大利面条,通常称这样的代码为"面条式(Spaghetti)"代码。后来就比喻程序代码跳转太多,结构不清晰,HTML 和 VB,ASP 对象或其他语言嵌在一起。绞来绞去像意大利面条似的。

"面条式"代码存在一个主要缺点:可读性差,很难确定它到底完成什么工作。许多程序开始还是结构化的,但性能优化后都变成了"面条式"代码;但是,实际上"面条式"代码很少是高效的,因为太难确定它到底完成什么功能,从而也很难使用更好的算法来改

进。因此，最后导致“面条式”代码比结构化代码的效率更低。

“面条式”代码的确可能会提高程序的性能，但这只是在对程序做完所有其他可能的改进还不能达到要求时，不得已而采取的办法。开始编写程序时，要尽量在程序中使用直接的 if 和 switch 语句，当程序正常工作并易于理解后，才开始合并代码段（使用 jmp 指令）。当然，除非值得做，否则不要消除程序的结构。

3. 级别 3：编程和回放

这个级别是多个被测软件版本有效使用测试自动化的第一个级别。测试经理需要在实际投资开始前确保测试团队和客户对项目有充分的信心和安全感。如果没有经过技术培训，测试人员将不具备到达这个级别的能力。因为在这个级别中，测试人员要很好地理解自动化测试工具所有测试功能，还要掌握测试脚本语言知识。

优点：确定了测试脚本的设计，在项目早期就可以开始自动化测试。测试人员能够在项目的早期就开始进行测试脚本的设计，与开发人员一起研究他们认为可能会存在问题的地方，确保了开发人员把精力放在设计得到可用于测试的方案上。

缺点：要求测试人员具有很好的软件技能，包括设计、开发等。

用法：大规模的测试用例被开发、执行和维护的专业自动化测试。

4. 级别 4：数据驱动的测试

对于自动化测试来说这是一个专业的测试级别。测试人员拥有一个强大的测试框架，这个测试框架能够基于根据被测试系统的变化快速创建一个测试脚本的测试功能库。维护成本相对比较低，而且测试中还会使用到大量真实的数据。

优点：测试人员能够维护和使用良好的测试数据，这些数据有效地模拟真实生活中数据。

缺点：要求测试人员具备软件开发的技能，能够访问相关的测试数据。

用法：可用于大规模的测试用例被开发、执行和维护的专业自动化测试。该级别对测试数据要求较高。一个测试人员要花费一些时间来识别在哪里收集数据和收集哪些数据。使用现实生活中的数据是最基本的，这些工作完成后，测试人员才能够通过使用现实的数据来运行大量的测试。使用良好的数据将为测试人员提供发现错误的能力，而这些错误通常在项目后期才会被发现或者被客户发现。

5. 级别 5：使用关键词的测试自动化

这是自动化测试的最高级别，主要的思想是将测试用例从测试工具中分离出来。这个级别要求有一个具有高技能的测试团队，这些测试人员能够将测试工具的知识与他们的编程能力结合起来。这个团队负责在测试工具中生成并维护测试方案，能够使测试工具从外部的来源，例如 Excel 表或者数据库中执行测试用例。这种测试概念最初是由 CMG 开发的。其他的开源方案有由 SAS Institute 的 Carl Nagel 开发的 DDE。使用 DDE 的概念，测试人员关注点放在 Excel 表中创建测试用例上面，保存和使用一些特定动作关键词。执行过程是从 Excel 表中读取测试用例，并将测试用例转换成为测试工具

能够理解的形式，然后使用不同的测试功能来执行测试。

优点：测试用例的设计被从测试工具中分离出来，测试人员可将关注点放在设计良好的测试用例上。允许测试用例的快速执行和基于用例的评估。

缺点：需要一个熟悉工具和具有开发技能的测试团队，以提供并维护测试工程（框架）。

用法：这种专业的测试自动化能够将技能的使用达到最优化。使用关键字驱动的测试框架，测试人员可以使用 Excel 来生成实际的测试用例。这个级别对于那些按照正规使用测试用例的组织或者项目来说是非常适合的。测试人员可以集中精力来生成第一个包含被需要“对象映射”的测试用例（主流程）。如果测试用例设计不错，需要做的工作也非常简单。根据测试应用的复杂程度，通常这会花费大约半天到一天的时间。后续的每一个测试用例大概会花费 15～20 分钟的时间，因为通常大多数测试用例可以复制已有的测试用例，并对其进行必要的修改，通常这种修改工作量不大。关键字驱动框架能够通过使用测试用例，使紧密的、并行的测试用例开发变为可能。

目前，大多数测试工具处于数据驱动到关键字驱动的阶段，有些工具厂商已经提出了支持关键字驱动的版本。

4.4 测试脚本技术

4.4.1 测试脚本分类

下面来讨论不同的脚本技术及用途。这些技术并不是相互排斥的，事实恰好相反，它们是相辅相成的，每种脚本技术在支持脚本完成测试事例的时间和开销上都有各自的长处和短处。应该注意到，对于软件测试来说，使用哪种脚本技术并不是最主要的，脚本所支持的实现测试事例体系的整体考虑才是最主要的。

1. 线性脚本技术

线性脚本是使用简单的录制回放的方法，录制手工执行的测试事例而得到的脚本。这种脚本包括所有的击键、功能键、箭头控制测试软件的控制键及输入数据的数字键。如果用户只使用线性脚本技术，即录制每个测试事例的全部内容，则每个测试事例可以通过脚本完整地被回放。

线性脚本也可能包括比较，如“check that the error message position riot valid is displayed.”。录制测试事例时，可以添加比较指令（如工具支持）或在回放脚本录制的输入时增加比较指令。但要知道手工运行 10 分钟的测试事例，而带比较的自动化测试可能需要 20 分钟到 120 分钟。因为当增加新的比较指令时，需要重新回放，新增加的脚本也应进行测试。应用和测试事例越复杂，这个过程花费的时间越多。

几乎任何可重复的操作，都可以使用线性脚本技术自动化。测试案例只用一次的或研究测试工具能否在给定环境中执行，也可采用线性脚本技术。

线性脚本可以用于演示或培训，例如演示的时候，希望向客户介绍软件功能，而又不希望不断地进行多少令人有点紧张的击键操作时，可以回放录制好的脚本代替击键操作。

2. 结构化脚本技术

结构化脚本类似于结构化程序设计，结构化脚本中含有控制脚本执行的指令，这些指令或为控制结构，或为调用结构。所有测试工具脚本语言支持顺序、选择、循环三种基本控制结构。

第一种形式为“顺序”脚本（即前面介绍的线性脚本），第一条指令第一个执行，然后执行第二条指令，以此类推。

第二种形式为“选择”控制结构，选择控制结构使脚本具有判断功能。最普通的形式就是 if 语句判断条件或真或假，例如检查特定消息是否显示在屏幕上，如果显示消息则继续进行，否则停止。

第三种形式为“循环”结构。有的脚本需要重复一个或多个指令，这时可用到“循环”结构。在这种结构中，指令序列按照指定次数重复运行或直到条件满足，例如从一个文件中读取数据记录，可以设计指令序列以某种方式读取和处理信息，然后重复这个指令序列，直到文件中的所有记录被读取和处理完。

除控制结构外，一个脚本可以调用另一个脚本，即将一个脚本的控制点转到另一个子脚本的开始，执行完子脚本后，再将控制点返回到第一个脚本，这种机制可以将较大的脚本分为几个较小的易于管理的脚本。

这种结构化脚本不仅可以提高脚本的重用性，还可增加脚本的功能和灵活性。充分利用不同的控制结构，可以开发出易于维护的脚本，更好地支持自动测试体系的有效性。结构化脚本技术的主要优点是健壮性更好，可以执行许多其他类似的功能，如需要重复的指令可以使用循环结构，还可以作为模块被其他脚本调用。其缺点是使脚本变得更加复杂，而且测试数据仍然“捆绑”在脚本中。

结构化脚本案例

这个脚本测试打开文件是否提示保存：

```
SelectOption"File/Open"
If message='The file has changed.Do you want to save the changes?'
Left MouseCheck 'Yes'
EndIf
Focus On 'Open'
Type 'filel'
LeftMouseClick 'Open'
```

这个脚本对“Do you want to save the changes?”消息进行检查。如果显示该消息，单击 Yes 按钮，否则继续执行下面的测试脚本。该脚本采用了 If 语句，其健壮性好，也可以对一些容易导致测试失败的特殊情况进行处理。此外，在结构化脚本中，使用循环语句，它可执行许多重复的操作。

3. 共享脚本技术

共享脚本是指脚本被多个测试事例使用,它可以在不同主机、不同系统之间共享脚本,也可以在同一主机、同一系统之间共享脚本。这种技术的思路,是产生一个执行某种任务的脚本。而不同的测试要重复这个任务,当要执行这个任务时,只需在每个测试事例的适当地方调用这个脚本。

这样将带来两个好处:第一,可以节省生成脚本(编写或录制指定的操作)的时间;第二,当重复任务发生变化时,只需要修改一处脚本。共享脚本技术的使用提高了自动化测试的效率和质量。共享脚本技术分为两种类型:一种是不同的软件应用或系统的测试之间共享脚本;另一种是同一软件应用或系统的测试之间共享脚本。

建立共享脚本的时间可能要长些,因为需要建立更多的脚本,且每个脚本需要进行适当的修改,才能达到脚本共享的目的。如果想从共享脚本技术中获得更多的收益,应该注重实践,确保所有的测试在适当的时候,能使用共享脚本。程序员一般在开发程序前,花点时间寻找一下是否有可用的函数,知道某个可重用的脚本是一回事,而查找这样的脚本又是另外一回事。如果不能很快找到脚本,可能需要自行编写,而且还将有更艰巨的维护修改任务。当脚本需要修改时,进行维护的人员可能认为仅共享脚本需要修改,而事实上还可能有其他的脚本也需要编辑。有的可能在运行测试时才意识到脚本需要修改,还有的甚至在出现问题时还不知道问题所在。某些形式的可重用脚本库有助于解决这个问题,但需要事先认真地建立和管理脚本库,脚本库也是测试体系结构的一部分。

共享脚本的另一个需要注意的方面就是脚本文档。这些脚本需要文档化,使测试者清楚每个脚本的功能,以及如何使用它们。如果软件文档不规范,文档又不及时更新,导致存在错误。如果这些问题带到脚本的文档中,那么自动化的实现过程和维护开销更大。解决这些问题的办法就是针对脚本制定规定或标准,这应该成为软件测试自动化体系中的一部分。不仅使文档具有参考价值,而且有助于指导新的脚本编写者建立脚本。

4. 数据驱动脚本技术

数据驱动脚本技术将测试输入存储在独立的数据文件中,而不是存储在脚本中。脚本中只存放控制信息,执行测试时,从文件中,而不是直接从脚本中读取测试输入。这种方法的最大好处是同一个脚本可以运行不同的测试。将数据文件单独列出,并选择合适的数据格式和形式,可使用户的注意力集中到数据的维护和测试上。达到简化数据,减少出错概率的目的。

例如,保险系统的一个测试是输入新保险策略的详情并验证数据库是否被正确修改。另一个测试除使用不同的保险策略外,进行相同的操作,因此需要相同的指令,但输入和期望输出不同(即描述不同的保险策略的值),这两个测试可以使用一个测试脚本和一个数据文件。

数据驱动技术的另一个优点是数据文件的格式,对于测试者更容易于处理。例如,对于复杂一些的脚本,数据文件中可以包含一些脚本运行时可以忽略的注释,而使得数据文件更易于理解,因而更容易维护。另一种方法是使用不同的格式说明测试输入。例如,很

多测试者常使用的电子表格软件。电子表格完成后，再转换为更自然的格式。电子表格文件作为主文件，任何修改直接对电子表格而不是数据文件。每当修改数据文件时，都是从电子表格中产生。这种方式的好处在于可以方便地选择测试数据的格式和形式，可以将更多的精力放在自动测试和维护测试上。在不同的数据格式和形式上花费一些时间，以发现一种易于测试者编辑测试数据的格式，这是值得的。数据格式越简单，越不容易出错。

测试的初始建立，需要花费一定的时间。但当有几百个测试而不是几十个测试时，所获得的收益将远远超过所花费的开销。当实现合理的综合测试脚本集时，可以完成和运行许多测试。在实际中采用这种方法的组织，所完成的测试要比手工运行的测试多得多。在一个软件发布范围内，还可以多次运行这些测试，大部分为回归测试。

如果测试不多，使用这种方法，开销可能大些。因此对于小系统而言，这个方法是不适合的。而对于大系统，生命周期较长，并且改动频繁，使用这个方法可以获得更多的收益。

5. 关键字驱动脚本技术

关键字驱动脚本技术，实际上是较复杂的数据驱动脚本技术的逻辑扩展。数据驱动技术的限制是每个测试事例执行的导航和操作必须一样。测试的逻辑“知识”建立在数据文件和控制脚本中，因此两者需要同步。然而，脚本的一些智能活动，不能移动到数据文件中。一个办法是允许控制脚本支持广泛的测试事例，而这大大增加了数据文件的复杂性（因为数据文件此时要包含脚本指令），这种代价很大，而且调试这种方法，实现的自动测试事例是十分困难的。

而关键字驱动脚本技术说明自动测试事例可以不用说明所有细节，将数据文件变为测试事例的描述。用一系列关键字指定要执行的任务，控制脚本可解释关键字，但这是在控制脚本之外完成的，要求一个附加的技术实现层。尽管听起来比较复杂，但运行起来要简单得多，关键是要正确标识关键字。

关键字驱动脚本的数量不随测试用例的数量变化而变化，而仅随软件规模而增加。这种脚本还可以实现跨平台的用例共享，只需要更改支持脚本即可。关键字脚本驱动技术主要应用于软件测试自动化的工程应用领域和数据库应用中。关键字驱动要求工具能提供尽可能简单的输入测试方法，以减少不必要的重复和琐碎，这样就可以在外部数据文件里维护检查点和执行操作的控制。因此测试数据和测试的操作序列控制都是在外部文件中设计好的。

例如，为完成在网页浏览时输入网址，一般的脚本需要说明在某个窗口的某个控件中输入什么字符，而在关键字驱动脚本中，可以直接在地址栏中输入网址，甚至更简单，仅说明输入网址。

简单的捕获回放并非测试自动化。测试工具录制产生脚本是不能直接使用的，必须在此基础上采取添加判断和循环等控制结构、提取公用代码以供复用、采用数据驱动策略使脚本和数据分离等措施。因此，开发自动化脚本并非像想象的那么简单，必须按照软件工程的方法对脚本开发进行计划和组织，这样才能使开发出来的脚本真正实用。

4.4.2 测试脚本应用

1. 脚本语言介绍

为了帮助读者掌握脚本语言，这里介绍两种脚本语言 GUI 脚本和 VU 脚本。

GUI 脚本即图形界面脚本，主要由 SQA Basic 语言编写，这是一种扩展的 Basic 语言，可利用图形界面来生成脚本，用来记录测试人员对被测软件的操作动作（鼠标、键盘）Windows 消息。通过回放这些消息来模拟测试人员的测试操作，当然也可以直接编写 GUI 脚本，通过自动化测试工具发出 Windows 消息来执行相应的图形界面操作，达到测试目的。通常的做法是先录制一段 GUI 脚本，然后根据测试人员的目的，在此基础上进行编辑，最后回放编辑过的脚本进行测试。

VU 脚本即虚拟脚本，主要由一种类 C 语言编写，用来记录 C/S 系统在网络上传送的命令和命令响应的网络消息，通过编辑并回放这些网络消息脚本，可以模拟虚拟用户和服务器系统的交互，所以 VU 脚本经常用于测试客户端应用程序与服务器、数据库以及 Web 服务器交互时的性能。

2. 脚本编写技巧

掌握以下测试脚本编写技巧，可以更好地开发高质量测试脚本。

（1）增量式开发脚本

为了可以成功地回放脚本，需要调试几百行的代码，为了参数化的数据驱动测试脚本能包含几个数据集，常见的调试测试脚本方法是首先录制所有的业务流程和需求，然后测试人员回放测试脚本以验证并纠正问题。测试人员继续调试脚本直到它可以和一组（或多组）数据集一起成功地回放。

当测试脚本有成百的代码行、验证点、分支的逻辑、错误处理、参数和数据存在于多个已录制的业务流程之间，调试、解决测试脚本中的问题变得特别的乏味和难以处理。为了能够录制和调试那些复杂且又冗长的测试脚本，一个好的方法是先录制脚本的一部分，并且在录制测试脚本的其他部分之前进行调试。在测试一部分脚本后，可以再与其他部分脚本合成测试，这样逐步增量测试。在测试脚本的所有部分都录制后，测试人员就可以回放整个测试脚本，并确保脚本同一个或多个数据集一起从头到尾被正确回放。

例如，录制并改写一个执行了以下业务流程的复杂的测试脚本。

① 检查在货仓中的库存；

② 执行一次 MRP 运行；

③ 补充库存；

④ 挑出一些要发送的货物进行发货；

⑤ 确定交货需要移交的订单；

⑥ 验证发送的货物到达了它们的目的地。

这个测试脚本有一些代码行、参数、验证点和需要，这些要保持数据相关性，使之作为

一个整体而存在。首先录制每一个单独的流程并且验证他们分别可以成功地回放。然后将所有录制好的流程集成一个大的测试脚本,并且验证它同多个数据集一起能够成功地回放。这个过程的关键在于确定在继续录制整个测试脚本的剩余部分之前,每一个已录制的流程可以成功地回放。

(2) 测试脚本的同步

测试工具会以比终端用户手工按键快得多的速度回放已录制的测试脚本。如果被测程序不能即时显示数据或从数据库取出数据,以允许测试脚本正确地回放,这可能会造成被测程序的崩溃。当被测程序不能响应测试脚本时,脚本执行会突然中断,然后需要用户干涉。为了所测试应用程序与回放中的测试脚本同步,测试小组在已录制的测试脚本中引入了人为的等待时间。为了放慢测试脚本的执行,嵌入在测试脚本中的等待时间是任意的,只是凭借自己的经验。等待时间主要的问题是不合拍,要么等的太长,要么就是时间不够。

例如,测试人员或许注意到所测试的应用程序测试脚本回放得太快。他可能打算放慢,尝试多次,直到测试脚本执行和测试的应用程序同步。这个技巧可能会造成相反的结果,甚至失败。如果在测试执行时,由于外部的因素(例如网络有延迟或系统维护)导致应用程序运行比新引入的等待时间更慢。在这种情况下,每次测试人员只好不断地猜测一个新的合理的等待时间。用等待时间减少脚本回放速度是不科学的,并且在没有用户干涉情况下,对于创建强健的自动化测试脚本也没有什么帮助。

测试人员应尽量避免引入人为的等待时间或任意的 sleep 变量,以使测试脚本和应用程序同步。引入 while 语句或嵌套的 loops 语句是一个很好的办法,它可不管被测程序的响应时间都可以使脚本成功回放,也可以减少在测试脚本回放时用户的干涉。例如插入 while 语句在录制好的测试脚本里,不断按 Enter 键,直到创建了一个计划中的协议,不管所测试应用程序要花多长时间产生协议。这样测试脚本就可以不依赖被测程序的响应时间而工作。

(3) 录制回放隐藏的对象

录制一个双击动态表格的某个字段的脚本,如果表格中字段的位置是变化的。那么脚本回放就可能失败。测试脚本在回放中失败很多情况下都是因为某些可见或不可见的对象位置发生了改变。为了让位置敏感的脚本能够正确回放,需要使用功能性增强脚本,例如在页面中点击"向下滚屏"、"下一页"或"查找"按钮。功能增强性脚本能够自动识别录制的对象,并在回放中正确地对其进行调用,而不用考虑这些对象的位置是否变化。

例如,录制一个脚本实现向下滚屏两次来查找一个可以在表格中输入的空字段。但几个星期后情况变了,需要向下滚屏四次来查找空字段,而不是此前录制的两次的脚本。如何改写脚本?

答案应在脚本中嵌入逻辑判断,以指导脚本向下滚屏需要的次数来查找一个空字段,通过在一个 while 循环中放置一个"下一页"(next page)功能实现这个目的,它可以驱动脚本不停地滚屏直到找到空字段。

(4) 为脚本创建消息通知

为了增强测试脚本的健壮性,可以给测试脚本添加错误处理的功能。当错误发生时,

发送错误信息给记录设备或某个预订邮件地址。这个发送消息通知的机制常见于夜间测试，因为一些重要的测试可能会安排在晚上执行，为了避免自动化测试失败，就需要正确地运行这些测试，并处理好运行结果，发送消息通知就显得很重要了。通常关键业务的测试脚本，都会有一旦出现失败就自动发送消息进行通知的功能。

(5) 实行版本控制

版本控制可以帮助追踪测试脚本中的变更，并可维护同一测试脚本的多个版本。还需要养成良好的编程习惯，例如添加注释。脚本中需要包含一个描述（如它是干什么用的）和特别用途（如回归测试）的文件头。脚本的文件头应该包括脚本的作者、所有者、创建和修改的日期、脚本可以追溯到的需求识别符，脚本所支持的业务范围、脚本中的变量和参数数量。还有其他有助于脚本执行修改和维护的相关信息，例如列出所有有依赖的测试脚本、脚本运行条件、需要在脚本运行过程中打开或关闭的应用程序、数据格式等。

其他技巧还有很多，这里就不一一列举。

3. 脚本技术的评价

由于脚本技术是测试体系的一个关键部分，因此保证脚本的质量是非常重要的。好的脚本应该满足要求，也就是说，好脚本应是可靠的，并且易于使用和维护。经验表明，可以通过一些特性或属性来描述脚本的易用性和可维护性。表 4-4 给出了对好脚本或差脚本进行评价的基本依据。

表 4-4　脚本评价依据

属　　性	好测试脚本集合	差测试脚本集合
脚本数量	少（少于一个测试事例一个脚本）	多（至少每个测试事例一个脚本）
脚本大小	小（包括注释，不超过两页）	大（单个脚本许多页）
功能	每个脚本有一个明确单一的目的	执行许多功能甚至是整个测试事例
文档	提供给用户或管理者的文档清晰、简洁	无文档或文档不更新，内容不详细
复用性	及时更新许多脚本，可以完成不同脚本的测试事例	无复用性，每个脚本只可以复用于完成单个测试事例
结构化	结构易于理解，容易修改，根据良好的编程经验，合理组织控制结构	组织混乱，修改困难
维护性	易于维护，软件的变化只需要对少数脚本做小的修改	较小的软件变化需要较大的脚本修改，并且修改困难

4. 注意事项

对于脚本的开发需要注意以下问题：

① 自动化覆盖率越高，前期消耗的工作量越大。要充分利用函数库，将很多的通用函数放进去，这样编写脚本时，可以大量调用已经存在的函数，这样工作量会小得多，效率也会提升很多。

② 需要对测试脚本进行维护。其实编写脚本的技术含量并不那么高，只要稍加培

训，绝大多数没有接触过的测试人员都能够完成任务。

③ 开发脚本必须遵循自动化测试标准，就类似于程序员编程规范一样。测试脚本就好比是测试人员的程序，同样要求编写规范。只有养成这样的好习惯，才能方便维护脚本，既避免增加后期的维护量，也方便使用者使用。每个脚本的文档名应放在脚本的开头，并且尽可能简洁。这两点对于修改脚本很重要，因为修改脚本最可能修改文档以及脚本内容。

④ 保证开发的脚本回放在没有问题的基础上，适当增加出错处理来增强脚本的健壮性。

⑤ 后期还可以在脚本中加入检查点，这样做的好处是可把原来需要人工去校验的地方让脚本去做。

⑥ 在脚本中增加数据驱动方法，使脚本能覆盖更多的分支路径，进一步提高脚本的集成度。脚本是不会执行那些没有被编写进去的功能点，所以说后期测试人员一旦发现某个地方可以让脚本来代替手工进行执行，就可以不断地增强自动化脚本。

4.5 自动化测试实践

衡量自动化测试的标准主要是测试的敏感性和健壮性。

① 敏感测试是指测试过程能发现微小的，甚至是不起眼的错误。敏感的测试能发现较多的问题，但任何一点微不足道的改动都将导致测试用例及执行过程的更改。

② 健壮测试是指测试过程能够容忍较多的差别，不会影响测试用例的失败。在自动化测试时，健壮的测试可以较容易通过执行，这样减少了意外情况对测试过程的影响，但会导致发现问题的能力下降，甚至放过应该被发现的问题。

设计测试时，应当在测试的敏感性和测试的健壮性之间进行权衡。对大量的自动测试用例而言，敏感性过剩比敏感性缺乏更容易对失败分析工作造成反面影响。如果运行一组敏感的测试用例，那么有可能其中多数的测试用例因为相同的原因而失败。在这种情况下，每个失败的测试用例都指向相同的错误，但测试人员希望获得的却是不同的错误及错误的差异，并不需要重复且相同的错误报告。

权衡使用敏感和健壮测试的原则：

① 软件发生了变化，即在进行回归测试时，使用敏感测试；

② 如果有多组测试用例，对其中的一两组使用敏感测试，而其他组使用健壮测试；

③ 好的测试自动化策略应该是根据实际情况混合使用敏感测试和健壮测试。

部分的测试工具可以实现测试用例的自动生成，但通常的工作方式为人工设计测试用例，使用工具进行用例的执行和比较。软件测试自动化的设计并不能由工具来完成，必须由测试人员进行手工设计，但是在设计时必须考虑自动化的特殊要求，否则无法实现利用工具进行用例的自动执行。为此，就必须在测试的设计和内容的组织方面采取一些特殊的方法。下面简述自动化测试的基本工作过程。

4.5.1 基本工作过程

1. 前期准备工作

在对一个软件系统进行测试之前，可以先改进被测试的产品，使它更容易被测试。有很多改进措施可以帮助测试人员更好地使用产品，也可以帮助测试人员更好地测试产品。例如一些产品很难安装，可以直接改进产品的安装程序，或者是开发一套自动安装程序，目前有很多专门制作安装程序的商用工具。

也可以利用工具在测试执行的日志中查找错误，利用手工查找报错的方法容易让人感到乏味且易遗漏。这时可以仔细了解日志中记录的错误信息格式，写出一个错误扫描程序，使查错自动化。

2. 需求分析

正如软件生命周期有需求分析阶段一样，在制定测试方案之前也需要收集需求。定义自动化测试项目的需求，要求全面地、清楚地考虑各种情况，然后给出权衡后的需求，并且可以使测试相关人员更加合理地提出自己对自动化测试的意见。

开发管理者、测试管理者和测试人员实现自动化测试的目标常常是有差别的。除非三者之间达成一致，否则很难定义什么是成功的自动化测试。为了避免这种情况，需要在自动化测试需求上保持一致。应该有一份自动化测试需求，来描述需要测试什么。测试需求应该在测试设计阶段详细描述出来，自动化测试需求描述了自动化测试的目标，也描述了自动化测试的内容，定义哪些需要自动化测试，哪些需要手工测试。

选择适合实行自动化测试的技巧

(1) 提取适合自动化的测试

首先，制订表格提取适合自动化测试的项目，这里的原则是挑选最能获得投资回报的测试项，表现在最能缩短时间周期、减少风险和提高测试精度。4.3 节 RUP 据此原则提出了适合实行自动化测试的 6 个测试类型。

(2) 评估每个自动化测试的时间消耗

目前没有简单的数学模型判断自动测试和手工测试的时间消耗比例。根据 RUP 测试专家 Cem Kaner 估算，开发一个自动化测试的时间，包括从创建、校验、文档化自动测试的时间消耗是手工测试同样过程的 3～10 倍，自动化测试专家 Linda Hayes 认为是 5～10 倍。对于复杂的测试，甚至更长。因此，一个需要 100 小时的测试套件，如果实行自动化测试，就需要 300～1000 小时或更多的时间。

估算毕竟是估算，必须根据企业测试人员的实际测试技能、测试软件的实际特征，以及测试工具的实际使用复杂度进行判断。但是有一点是毫无疑问的，就是初次实施自动化测试的时间消耗，要比熟悉工具和测试流程需要的时间更长。因此在评估自动化测试的时间消耗时，一定要将其考虑在内。例如，一个 1600 个测试用例的项目，估计前 400 个用例每个需要 4 小时，接着 400 个可能每个只需要 2 小时，最后的 800 个，每个将只需要

1小时，故而全部时间是：1600小时+800小时+800小时=3200小时。基本原则是，挑选时间消耗比例大的测试，优先实行自动化测试。

(3) 根据测试目标确定自动化测试的优先顺序

最后，确定自动化测试的优先顺序。这里最重要的原则就是采用迭代的方式，确定自动化测试的执行顺序。首先确定每个迭代的目标，挑选最能获得投资回报的测试，例如冒烟测试几乎总是能立即获得时间和资源上的回报；再挑选最容易开发的脚本、最容易理解的测试实行自动化，之后逐渐扩展并迭代。

3. 自动化测试工具的选择

有的可以根据要测试的软件系统，自己开发一个测试工具。但它的花费代价较大，鉴于此，可以选择已成熟的测试工具。对于测试工具的选择，要有专人针对不同的自动化测试，去评估究竟该使用哪种测试工具比较好。自动化测试工具又分单元测试工具、功能自动化工具和性能自动化工具，其中又分开源的工具和商业的工具。究竟哪种工具更适合自己平台的测试，还需要专业人员进行评估。

注意：自动化测试工具的引入是一个需要详细考虑的问题，不要为了应用工具而进行自动化测试，工具是为了自动化测试而产生的，有时候可能完全失效，因为工具不可能满足和适应所有软件的需求，此时，就需要测试人员自己动手编写程序或脚本来实现自动化。

4. 制定明确的自动化测试计划

RUP提出计划就是投资。计划管理是自动化测试的关键实现，在一个测试项目里，虽然紧缩或干脆忽略掉制订计划的过程，是相当具有诱惑力的事情。尤其在项目周期短、时间紧的情况下，如果没有明确的计划，尤其那些初次实施自动化测试的软件企业，尽管在初期可以稍稍尝到自动化测试的甜头，但最终根本无法体现自动化测试的种种优势，而且很难成功。因此，针对自动化测试项目，一定要制订明确良好的计划。如果对自动化测试不做时间、资源上的计划安排，自动化测试不仅很难成功，最终还要消耗和浪费的不仅是自动化工具，还有人力、物力和金钱。

自动化测试计划包括指明测试中需要什么样的数据，并给出设计数据的完整方法；需要明确测试设计的细节描述，还应该描述测试的预期结果；完成测试设计文档，需要描述清楚测试设计的思路；编写测试脚本。

计划应与时俱进。最初的计划无法覆盖全部内容。可能还没有谁第一次做的计划可以涉及全部内容，也没有日后不经修改的完美计划。初次的计划必将为今后提供参考。一般来说，初次的自动化测试计划内容包含选择那些易于维护和复用的基本功能结构，然后对其编写测试脚本。从这个意义来说，意识到在初始阶段对自动化测试计划的投入，必将对将来测试项目产生深远的影响。

测试用例案例

一个查看网上订单的测试用例实例：

(1) 测试用例描述

查看客户订单，订单包含唯一的标识符、状态、归属人、订单组成、订单数量、总金

额等。

(2) 执行条件

① 前置条件：客户登录系统不具有管理员权限，Classics Online 主窗口打开，两个数量为 1 的不同订单显示在客户名下，一个买的是 Brandenburg，另一个是小提琴。

② 测试输入：从订单菜单里选择"View Existing Order Status… "，查看客户订单后，关闭查看窗口。

③ 观察点：View Existing Order 窗口弹出，订单的客户名显示正确，其他列的抬头显示 ORDERID、STATUS、COMPOSER、COMPOSITION、QUANTITY、TOTAL 等，滚动条可以正常滚动。

④ 控制点：在 View Existing Order 窗口里，具有关闭按钮和 X 按钮以关闭该窗口，还有 Cancel Selected Order 按钮。

⑤ 期望结果：两个订单有两个不同的标记数字，状态是 Order Initiated，归属人是 Bach，购买物是 Brandenburg 和小提琴，数量各为 1，总金额分别是 18.99 元和 16.99 元。

⑥ 后置条件：测试完成后，Classics Online 主窗口自动激活。

测试用例的书写要有相对固定的模板和表现形式，并存储在固定位置以方便执行和跟踪，测试人员也必须严格按照测试用例执行测试或开发测试脚本。

5. 可行性分析

可行性分析是指验证自动化测试项目的可行性，尽快地验证采用的测试工具和测试方法的可行性，站在产品的角度验证产品采用自动化测试的可行性。同时要确定测试工具和测试方法对于被测试的产品和测试人员是否合适。选择一个快速、有说服力的测试套件，它是评估测试工具的最好的方式，可以证明所选测试工具和测试方法的正确性。

6. 自动化测试的维护和扩充

自动化测试是一个长期的过程，为了与产品新版本的功能和其他相关修改保持一致，自动化测试需要不停地维护和扩充。而且在自动化测试设计中，应考虑自动化未来的可扩充性，还要考虑自动化测试的完整性，这些很关键，也很重要。

编写测试计划技巧

一个稳定的测试计划可以分解测试的复杂性，并减少测试的风险。一般使用 Microsoft 的 Project 软件来编写测试计划。优秀的测试项目计划包括：

① 以简单术语描述测试项目；

② 计划项目的日程安排；

③ 评估项目的风险；

④ 设置项目的沟通计划；

⑤ 确定项目的所需资源。

在此重点讲述后四项内容。

日程安排：良好的日程安排计划对任何测试项目都非常重要，尤其初次实施自动化

测试，必须考虑构建自动化测试的时间。项目计划的日程安排应考虑如下方面：

- 对项目成员进行自动化测试工具和脚本开发的培训；
- 创建自动化测试的基本架构，包括定义测试流程、开发标准、创建自动化测试框架等；
- 计划和设计测试用例；
- 开发测试脚本；
- 创建并执行测试套件；
- 评估输入（需求、测试用例、测试策略等）；
- 配置测试环境（硬件、软件、环境等）。

风险管理：自动化测试项目计划的风险包括时间、资源、人力、项目细节等风险。这四项风险是概括性描述，实际项目中的风险还要根据实际情况细化并评估，然后写出避免风险的预备方案，如人员中途离职、需求变更造成的时间变更等。

沟通计划：由于项目成员间的经验偏差，以及交流能力的不足，经常会出现发生问题时不知该找何人负责的麻烦。因此，有必要在自动化测试计划里规定沟通计划的内容。所谓沟通计划，就是事先在计划里预计测试实施中出现的各种问题，然后指定每种问题出现时候的责任人，避免到时发生时，互相推诿。

资源计划：测试项目所需资源包括两方面：

- 人力资源：人数、工作时间。
- 设备资源：硬件（服务器、客户端等）、软件、预备时间。

如果自动化测试程序报告测试用例执行通过，按理可说明执行通过的测试用例没有问题。但实际上并不完全是这样的，有很多存在问题的测试用例也能执行通过，虽然表面上通过了，实际上却执行失败了，而且没有记录任何错误日志，这是失败的自动化测试。这种失败的自动化会给整个项目带来灾难性的后果，而当测试人员构建的测试自动化采用了很糟糕的设计方案或由于后来的修改引入任何错误，都会导致这种失败的测试自动化。失败的自动化通常是由于没有关注自动化测试的性能或没有充分地进行自动化测试设计导致的。

4.5.2 开展自动化测试

1. 获得管理层支持的必要性

企业实施 RUP 测试，不是单纯测试部门的事情，更不是几个测试工程师单靠对测试工具的强烈兴趣，就能够在企业内部推广使用。有数据表明，很多测试项目的失败，并非技术的限制，更多的是缺乏企业管理层的支持。管理层的支持与否可以瞬间中断一个项目，而且没有管理层的支持，购买工具、测试环境与资源的花销，根本无从实现。而且推广自动化测试，势必影响企业内部软件的开发流程，试想没有高层的审批，实施工作根本无从下手。因此，为最大限度地保证 RUP 测试的实施，花费一定的时间，获得管理层的支持和必要的项目资源是非常必要的。

2. 正确看待自动化测试项目

绝对不要把自动化测试简单地看作是运用一套自动化测试工具的过程。应该把实施自动化测试的软件看成一个项目，并且把自动化测试项目看成一个企业新的里程。这个新里程有两个要素：

① 开创里程——确定自动化测试的涉众；

② 维持里程——改进组织管理过程以适应自动化测试。

3. 自动化测试的涉众

实施软件测试自动化，必须获得涉众的支持，这也是自动化测试涉众的根本任务。那么，需要获得那些涉众的支持呢？

（1）企业高层领导

从企业的高层领导获得：

① 自动化测试的可信度；

② 对测试工具、培训方面的财务支持；

③ 企业其他部门人员的支持，如审批、招聘等。

在和企业高层领导交涉时，应该如实介绍自动化测试的情况，说明自动化测试并非一定获得丰厚的投资回报，也并非立即能获得回报，并从企业角度设定切实可行的期望目标，例如只是在某类软件项目的某种测试类型或阶段实施自动化测试。

（2）测试主管

测试主管或经理直接监督企业整个测试过程的实行，并确定测试日程、战略、资源分配及工作细节，故而有必要获得测试主管对自动化测试的支持。

在和测试主管或经理交涉时，要说明如何使测试工作更加有效，让他们清楚自动化测试的功效，还要让他们通晓如何计划、实施自动化测试项目等。

（3）测试人员

和测试人员沟通，因为一旦实施了自动化测试，必将改变测试人员的原有工作方式，需要他们学习新的技能，与开发人员之间也要保持更紧密的合作，另外也需要他们严格遵守新的测试流程和规范。而且需要测试人员理解自动化如何提高工作效率，并清楚遵守测试流程的必要性，还要明确自动化测试和手工测试的平等关系，并非所有人都要成为自动化测试专家，自动化测试也无法完全取代手工测试，以免造成不必要的心理失衡。

（4）开发人员

获得开发人员对自动化测试的支持是非常关键的，需要鼓励开发人员开发优质的代码，增强软件的可测性，并通过有效沟通提高测试的覆盖率。另外，RUP 提倡开发人员执行每个发布版本的冒烟测试。

4. 测试规范的制订

自动化测试规范是企业高层对该方案的授权见证，同时加强与企业其他部门的交流与合作。没有该规范，遇到问题时会手足无措。例如有些自动化测试负责人，缺少执行的

可信度，对于自动化测试执行人，可能会遭遇各种阻力。还有企业没有清晰明确的测试目标和方案，各部门制订各自的规范，也必将在实行时发生冲突，从而导致项目的最终失败。

阅读材料——国内测试自动化的问题

尽管越来越多的具备优秀的实施自动化测试经验的企业陆续出现，仍然有些企业对自动化测试工作感到困惑。当前国内软件企业实施或有意向实施测试自动化时所面临的一些想法和问题：

- 认为测试自动化是个遥不可及的事情，小公司不必实施，因为人员、资金、资源都不足。
- 热血沸腾地实施测试自动化，购买了工具，推行了新的测试流程。几个月工具放在那里成了共享资源，测试流程又涛声依旧，回到原来的模式。
- 公司实施了自动化测试，然而开发与测试之间，甚至与项目经理之间矛盾重重，出了事情不知如何追究责任，虽然还在勉强维持自动化测试，但实施的成本比手工测试增加，工作量比从前更大了，从而造成项目团队人员怨声载道，更怀念起那段手工测试的日子。
- 自动化测试实施相对比较成功，但或多或少还有些问题，例如工具选择不准确，培训不到位，文档不完备，人员分配不合理，脚本可维护度不高等，造成一种表面上的现象，这种自动化测试流程是一幅空架子。

出现这些问题，要正确认识实施软件测试自动化所出现问题的根本原因：

- 目前国内的软件公司，很多还处于获取资本的原始积累阶段，不能说哪一个公司领导不重视测试，而是测试整体行业都没有被重视起来，这是其一；其二在公司高层看来有比这一工作更需要重视的环节，例如寻找客户签订单或开发，这些是直接关系公司存亡的命脉。
- 即便企业重视测试，但要量体裁衣。如果公司做一番比较全面地评估后，觉得不一定非要实施自动化测试。尤其是一些中小软件公司在大刀阔斧推行自动化测试之前，不如在测试流程管理、测试缺陷流程、测试人员技能培训等方面做工作，这样可以用比较少的成本投入来获取相对较大且长期的收益回报。
- 已经开展了自动化测试的企业，应该客观、认真地分析自动化测试没有达到理想效果的原因，并认真协调运行过程中出现的矛盾和问题。任何事情都有利弊，工作中扬长避短、兴利抑弊。

4.5.3 主要问题

自动化测试的优势很多，很多单位都引入了自动化测试工具，实施自动化测试。但是在实施过程中却普遍存在很多问题，导致整个测试工程失败。常见的问题就是认为自动化测试工具的利用率越高，从中获取的投资收益就越大。实际上从自动化测试工具获得的收益应该体现在测试质量上，而不是数量上，选择哪些测试实行自动化，如何开发执行测试脚本，要比单纯追求实行自动化测试的数量更重要。

此外还存在以下问题：

（1）对自动化测试期望值过高

认为所有的测试工作都应该实现自动化，自动化测试能发现很多错误。实际上100％的自动化测试是一个理论上的目标，要全部实现自动化，代价十分昂贵。另一方面，发现错误的数量主要是与测试用例的设计有关，与具体用什么测试工具关系不是很大。可通过定义需求来解决对自动化测试期望值过高的问题。自动化测试需求描述了自动化测试的内容，定义哪些需要自动化测试，哪些需要手工测试。

（2）轻视系统分析与设计

认为自动化测试就是录制/回放脚本，不需要系统进行分析与设计，只需要录制测试脚本。例如GUI测试，产品界面的改变对脚本的正常运行影响较大。如果不在开始对系统进行很好的设计，很容易导致在界面发生改变以后，以前的测试脚本全部作废。所以在项目开始初期，应利用面向对象的方法或对其项目进行分析与设计，保证项目的可扩展性和组件的重用性。这个时候可以利用RUP的实践经验——基于组件的体系架构来解决这个问题，它为系统设计指明了方向。设计模板告诉设计者怎样处理最重要的问题。没有一个坚实的架构或设计模板，就不会有项目大幅度的前进。

（3）忽视脚本质量对测试的影响

默认开发的测试脚本不会有错误。其实测试脚本本身就是程序，所以就有可能产生错误。很多时候测试项目时发现的错误就是测试脚本的问题。所以一定要重视测试脚本的质量，保证其正确性。RUP推荐使用持续的质量验证来保证测试脚本的质量。

（4）忽略需求变化对测试的影响

测试的目的就是为了保证产品的质量，满足用户需求。如果发现最后的产品偏离了用户的需求，这个问题就很棘手，测试也变得没有意义。在实际的项目中，很少有需求在整个项目固定不变的情况。因此需求的变化对测试的影响很严重，应重视这个问题。RUP推荐使用管理项目需求和迭代开发来控制需求变化对测试的影响，在开发周期内进行需求管理。这意味着项目需求会被反复和逐渐地确定、证明、评估和改进。

（5）只关注短期效益

企业购买自动化测试工具后，通常的做法都是将工具分派给相应部门，并立即着手创建、执行测试，虽然通过录制/回放脚本可以获取短暂的受益和喜悦，但是从长期来看，自动化测试的真正收益来自于脚本的重用，而这根本不是靠简单的录制/回放就能获得的。虽然获取计划的时间和资源比较困难，但是也要投入一定的时间和精力，以获取长期回报。越早投资于当前自动化测试项目的计划过程，就会从将来的项目中获取更大的收益。

4.5.4 建议

如果软件企业有意向实施自动化测试，那么应该具备什么样的条件才可以引入自动化测试呢？才可以最大可能地减少引入风险，并能够可持续性地开展下去呢？首先要对企业自身现状进行评估分析。

第一，从企业规模上来说，没有严格限制。无论公司大小，都需要提高测试效率，希望

测试工作标准化，测试流程正规化，测试代码重用化。所以第一要做到的，就是企业从高层 CTO 开始，直到测试部门的任何一个普通工程师，都要树立实施自动化测试的坚定决心，不能抱着试试看的态度。一般来说，一个这样的软件开发团队可以优先开展自动化测试工作：测试与开发人员的比例合适，如 1∶1～1∶1.5，开发团队总人数不少于 10 个。当然，如果公司只有三五个测试人员，要实施自动化测试绝非易事；不过可以先让一个、两个测试带头人首先试着开展这个工作，不断总结、不断提高，并层层向上级经常汇报工作的进展情况，最后再决定是否全面推行自动化测试。

第二，从公司的产品特征来说，一般开发产品的公司实施自动化测试要比开发项目的公司条件要优越些。原因很简单，就是测试维护成本和风险都小。产品软件开发周期长，需求相对稳定，测试人员可以有比较充裕的时间去设计测试方案和开发测试脚本。而项目软件面向单客户，需求难以一次性统一，变更频繁，对开发、维护测试脚本危害很大，出现问题时一般都以开发代码为主，很难照顾到测试代码。但决不是说做项目软件的公司不能实施自动化测试，当前国内做项目的软件公司很多，有一些正在推行 CMM 等级标准，只要软件的开发流程、测试流程、缺陷管理流程规范了，推行自动化测试自然水到渠成。

第三，要有标准化的开发和管理流程。很多软件公司采用了 CMM，也可能采用了 ISO，其流程也可能参考了 RUP。但不管是哪个标准体系，所有 IT 人员，甚至任何人员都应遵守以下工作原则：

- 把想做的写下来（计划管理）；
- 按照写下来的去做（行为管理）；
- 把做的事情记录下来（报告管理）；
- 出现的问题要设法解决（跟踪管理）。

如果软件开发团队据此开发软件，那么完全具备实施自动化测试的条件。当然，也许一些公司的测试管理比较混乱，出了问题不知道谁负责，测试人员或开发人员整日忙忙碌碌却无为，软件缺陷不胜枚举，那么这些公司还是要首先从管理角度来规范一下公司的开发流程和测试流程。

第四，从测试人员个人素质和角色分配来说，除了有一个 CTO 级人物做后盾外，还应该有个具有良好自动化测试背景和丰富自动化测试经验的测试主管，不仅在技术方面，更重要的是在今后的自动化测试管理位置起领军作用。还要有几个出色的开发经验良好的测试人员，当然也可以是开发工程师，负责编写测试脚本、开发测试框架；他们不需要对产品业务深刻了解，但要具备将软件业务逻辑转化成可测试逻辑的分析能力，便可以胜任自动化测试设计者。还有一些测试执行者，他们要对软件产品业务逻辑相当熟练，配合测试设计者完成设计工作，并在执行自动测试时，敏锐地分析和判断软件缺陷。如果你的测试团队具有这样的人员角色雏形，那么也就具备了实施自动化测试的又一条件。

综上所述的四个条件，企业可以决定是否推行自动化测试，但是为了减少实施风险，还要预测到其他潜在的风险，防范于未然。

阅读材料：对企业推行自动化测试的风险分析

(1) 资金风险

虽然你的公司具备实施自动化测试的条件，但如果企业效益不好，还是先扭亏为盈

吧。一款正版的测试工具价格昂贵，企业要首先考虑资金是否允许购买正版的测试工具软件，所以进行测试工具的成本估算，以及引入自动化测试后组织结构调整等方面的成本估算是很有必要的。如果你的公司处在如同前面所说的自动化测试试验阶段，可以使用试用版测试工具。当然具有实力的公司可以按照自身的工作流程自主开发测试工具，本文不考虑此种情况。

(2) 自动化测试对软件功能类型的切入点的风险

企业开发的产品业务和功能是否需要自动化测试，包括白盒自动化测试、功能自动化测试和性能自动化测试。例如一些公司开发单机版软件，只需要做功能测试，那就不必考虑性能自动化测试；有的公司开发简单界面之类的软件，例如搜索引擎，也不必考虑功能自动化测试；而大多数国内公司开发的软件，由于种种原因，一般都不考虑白盒自动化测试。也有可能公司开发的软件特殊性很强，市场上根本没有支持它的自动化测试工具，此时要另辟蹊径。这种评估相当重要，要根据自身的产品功能特征来综合评估。针对不同阶段采用自动化测试的种种优势，表 4-5 供读者参考。

表 4-5　不同阶段采用自动化测试的优势

测试阶段	描　述	备　注
单元测试/组件测试	通常是开发人员的职责，很多不同的方法能够被使用，例如"测试先行"，它是一个测试框架，开发人员在编写代码前编写不同的单元测试，当测试通过时，代码也被完成了	通过使用正式的单元测试，不仅能够帮助开发人员产出更加稳定的代码，而且能够使软件的整体质量更加好
集成测试	测试工作集中在验证不同的组件之间的集成上	测试通常是被测试系统的更加复杂测试的基础，大量的边缘测试被合并以制造出不同的错误处理测试
系统测试	通过执行用户场景模拟真实用户使用系统，以证明系统具有被期望的功能	这里不需要进行自动化的测试。安装测试、安全性测试通常是由手工完成的，因为系统的环境是恒定不变的
回归测试	重复已经完成过的测试，如果是手工完成的测试，通常只在项目的结尾再执行一到两次	有潜力应用自动化的测试，能够在每次构建完成后执行自动化的回归测试，以验证被测试系统的改变是否影响了系统的其他功能
性能测试	包括负载测试、压力测试、并发测试等	如果没有自动化的测试工具，你将不能执行通过模拟用户的负载实现的高密集度的性能测试

(3) 软件自动化测试切入方式的风险

正如前面所说，一定要将自动化测试与手工测试结合起来使用，不合理的规划会造成工作事倍功半。开始可规划自动化测试率的目标按 10%的自动化测试和 90%的手工测试来规划。如果这些目标实现了，再将自动化测试的使用比率提高。

(4) 企业软件的开发语言风险

当前业界流行的测试工具有几十种，相同功能的测试工具所支持的环境和语言各不相同，读者可以查阅更多的资料，去详细了解更多的工具。

(5) 时间估算

在评估完前面几项指标后，需要估算实施测试自动化的时间周期，以防止浪费不必要的时间，减少在人员、资金、资源投入上的无端消耗。虽然到测试自动化步入正轨以后，会起到事半功倍的效果，但前期的投入巨大，要全面考虑各种因素，做好周密的实施计划，并按计划严格执行，最大限度降低风险。

(6) 工作流程变更风险

测试团队乃至整个开发组织实施测试自动化，或多或少会因为适应测试工具的工作流程，带来团队的测试流程、开发流程的相应变更，而且如果变更不善，会引起团队成员的诸多抱怨情绪，所以应该尽量减少这种变更，并克服变更中可能存在的困难。

(7) 人员培训与变更风险

简单而言，就是测试团队人员的培训具有风险性，例如每个角色的定位是否准确，各角色人员对培训技能的掌握程度是否满意，尤其实施途中如果发生人员变更等风险，都要事先做出预测和相应的处理方案。

一个企业或软件团队实施测试自动化，会有来自方方面面的压力和风险，但是凭借团队成员的聪明才智和公司高层的大力支持，事先做好评估，做好风险预测。如果测试部门有意向引入自动化测试，那么首先要从思想上统一认识。

自动化测试能大大降低手工测试工作，但决不能完全取代手工测试。完全的自动化测试只是一个理论上的目标，实际上想要达到 100% 的自动化测试，不仅代价相当昂贵，而且操作上也几乎是不可能实现。一般来说，利用自动化的程度达到 40%～60%已经是非常好的了，达到这个以上将过大地增加测试相关的维护成本。

有些测试完全没有必要进行自动化，因为同一个项目自动化测试所需的时间比手工运行全部次数所需的时间总和还长。例如，手工运行一个测试要 10 分钟，而且一般每个月运行 1 次，那么一年需要 120 分钟。但如果自动化测试则需要 10 小时，那么这个测试需要连续不断运行 5 年才能收回成本。另外还应该注意避免自动化太多的测试。太多的工作导致参与人员工作积极性下降，可维护性下降，增加工作的风险。寻找可快速制胜的测试，尽快让大家看到工作成果，有助于获得更多的工作支持。

自动化测试能提高测试效率，快速定位测试软件各版本中的功能与性能缺陷，但不会创造性地发现测试脚本里有没有设计的缺陷。测试工具不是人脑，要求测试设计者将测试中各种分支路径的校验点进行定制，没有定制完整，即便事实上出错的地方，测试工具也不会发觉。因此，制订全面、系统的测试设计工作是相当重要的。

自动化测试能提高测试效率，但对于周期短、时间紧迫的项目不宜采用自动化测试。推行自动化测试的前期工作相当庞大，将企业级自动化测试框架应用到一个项目中也要评估其合适性，因此决不能盲目地应用到任何一个测试项目中，尤其不适合周期短的项目，因为很可能需要大量的测试框架的准备和实施而会被拖垮。

实施测试自动化必须进行多方面的培训，包括测试流程、缺陷管理、人员安排、测试工具使用等。如果测试过程不合理，引入自动化测试只会给软件组织或项目团队带来更大的混乱，如果组织或者项目团队在没有关于应该如何做的情况下实施自动化测试，那必将以失败告终。

4.6 自动化测试的优缺点

手工测试和自动化测试也是很多测试人员争相讨论的两种测试方法。有人对自动化测试趋之若鹜,也有人对自动化测试嗤之以鼻。如何看待自动化测试,首先要对两种测试有一个清晰的认识才能评说。

自动化测试是软件测试发展的一个必然趋势。好的测试自动化可以比手工测试达到更有效、更经济的效果。测试自动化的主要优点有:

(1) 软件新版本进行回归测试的开销最小

回归测试是测试自动化最主要的任务,对于软件开发,每发布一个新版本,其中大部分功能和界面都和上一个版本相似或完全相同,这时要对新版本再次进行已有的测试,这部分工作多为重复工作,特别适合使用自动化测试来完成,从而令回归测试的开销达到最小。

(2) 可以在更短的时间内完成更多的测试

基于计算机的高效计算能力,自动化测试最根本的优点在于与手工测试相比,能在更少的时间内完成更多的测试工作,因此也就缩短了测试时间。

(3) 可以完成一些手工测试不能或难以完成的测试

对于一些非功能性方面的测试,如压力测试、并发测试、大数据量测试、崩溃性测试等,这些测试用手工测试很难,甚至是不可能完成的。但自动化测试则能方便地执行这些测试,例如并发测试,使用自动化测试工具就可以模拟来自多方的并发操作了。

(4) 具有一致性和可重复性

由于每次自动化测试运行的脚本是相同的,所以对于自动重复的测试可以重复多次相同的测试。这样就可以获得测试的一致性,这在手工测试中是很难做到的。有些测试可能在不同的硬件配置下执行,使用不同的操作系统或不同的数据库,此时要求多平台产品的跨平台质量的一致性,这在手工测试的情况下更不可能做到。

好的自动测试机制还可以确保测试标准与开发标准的一致性。例如,此类工具可以测试每个应用或程序的相同类型的功能以相同的方法实现。

(5) 更好地利用资源

将烦琐的测试任务自动化,可以使测试人员解脱出来,将更多的精力投入到测试案例的设计和必要的手工测试当中。并且理想的自动化测试能够按计划完全自动地运行,这样就可以利用周末和工作日晚上的时间执行自动测试。

(6) 增加软件信任度

总之,通过较少的开销执行更彻底的测试,提高软件质量,这是测试自动化的最重要的一点。当然,自动化测试不是万能的,所完成的测试功能也是有限的。在软件版本还没有稳定的情况下,千万不要开展自动化测试,否则是自讨苦吃。

自动化测试存在着缺陷,体现在:

(1) 手工测试比自动化测试发现的缺陷更多

自动化测试的最大特点在于适合重复测试。一般情况下,以前运行过的测试再次用

来检查软件的新版本往往暴露的缺陷要少得多。据报道，自动化测试只能发现15%的缺陷，而手工测试可以发现85%的缺陷。

（2）自动化测试对测试质量的依赖性极大

通过自动化测试，实际上仅意味着测试的结果与期望值相同，因此测试的有效性很大程度是依赖于自动化测试本身的质量。确保测试的质量往往比自动化测试更为重要，所以要对测试软件进行必要的检测。

（3）自动化测试不能提高有效性

自动化测试并不会比手工运行相同测试更有效，尽管可以提高测试效率，但也可能对测试的进展起反作用。

（4）自动化测试可能会制约软件开发

应用软件的变化对自动化测试的影响要比手工测试更大一些，软件的部分改变有可能使自动化测试软件崩溃。而设计和实施自动化测试要比手工测试开销大，并需要维护，所以对自动化测试影响较大的软件修改可能受到限制。

总之自动化测试有很强的优势，借助计算机的计算能力，可以重复地、不知疲倦地运行，对于数据能进行精确的、大批量的比较，而且不会出错。但自动化测试不可能完全替代手工测试，因为很多数据的正确性、界面是否美观、业务逻辑的满足程度等都离不开测试人员的人工判断。而仅依赖手工测试的话，则会让测试过于低效，尤其是回归测试的重复工作量对测试人员造成了巨大的压力。因此可以得出一个结论：手工测试与自动化测试必须有机地结合，携手同行，要因地制宜地选择测试手段，充分利用各自的优势，为测试人员查找 Bug 提供各种方法和手段。

4.7 小　　结

本章介绍了软件测试自动化的专业术语及其特点，并且将手工测试和自动测试在技术上的不同进行了对比，揭示了两者之间相互联系以及在测试上的作用，分析自动化测试的受限条件。从不同角度对自动化测试进行研究，包括测试工具、脚本语言、代码分析技术等，归纳总结出自动化测试的一般过程和使用技术，对一般过程的具体环节给予了详细说明和解释。

并重点介绍了脚本技术，脚本技术可以分为以下几类：

线性脚本——录制手工执行的测试用例得到的脚本。

结构化脚本——类似于结构化程序设计，具有各种逻辑结构（顺序、分支、循环），而且具有函数调用功能。

共享脚本——指某个脚本可被多个测试用例使用，即脚本语言允许一个脚本调用另一个脚本。

数据驱动脚本——将测试输入存储在独立的数据文件中。

关键字驱动脚本——数据驱动脚本的逻辑扩展。

脚本中包含测试工具中使用的数据和指令，包括同步、比较信息以及从哪读数据和将

数据保存到何处以及控制信息。如 if 语句或循环结构。脚本技术类似于编程技术。合理的编程开发出的软件易于维护，同样合理的脚本产生的测试软件也是易于维护。脚本需要工程化。

通过分析当前国内软件企业的测试团队现状，并从管理角度分析测试团队实施软件测试自动化时所应考虑的问题，并从几个角度分析实施的风险，最终才可以成功引入自动化测试。为了不至于做事只做表面，每个测试团队中都必须要有专人去负责推动自动化工作的开展。还必须有专人负责维护脚本，规范脚本，甚至可以引入配置管理工具来统一管理脚本，把经验文档化。只有这样才会从中不断积累测试经验，也只有这样自动化测试效果才会更理想，也才能走得更远。

习题与思考

1. 简述自动化测试的定义。
2. 简述复用及模块化的原则。
3. 测试自动化可分为哪两类？
4. 根据 RUP 原则，优秀测试过程应具备哪些要素？
5. 为什么需要确定自动化测试项目的标准？
6. 自动化测试工具按执行的功能可分为哪几类？
7. 什么叫脚本？
8. 按照成熟度自动化测试可以划分为哪几个级别？并分别描述。
9. 描述自动化测试的基本工作过程。
10. 目前自动化测试存在哪些问题？如何解决？
11. 描述自动化测试的优缺点。

第 2 部分

单元测试

每个人都同意,是的,该做更多的测试。这种人人同意的事情多着呢,是的,该多吃蔬菜,该戒烟,该多休息,该多锻炼……这并不意味着我们所有人都会这么去做,不是吗?

但单元测试却远远不止是上面这些——也许你会认为单元测试是花菜那一类的,而我要说它更像一勺美味的调料,它让每份菜肴品尝起来更加可口。

——Andrew Hunt(《程序员修炼之道》畅销书籍作者)

Andrew Hunt 的比喻形象地说明了单元测试对于软件开发的重要作用。根据 RUP 理论,单元测试在迭代的早期实施,侧重于核实软件的最小可测试元素。单元测试通常应用于实施模型中的组件,核实是否已覆盖控制流和数据流,以及组件是否可以按照预期工作。这些期望值建立在组件参与执行用例的方式的基础上。开发人员在单元开发期间执行单元测试。实施工作流程对单元测试做出了详细描述。

单元测试的目的是保证软件开发的最小模块的质量,花最小的代价消除开发中存在的缺陷,从而让今后的工作变得更加轻松。单元测试应该由程序员自己来完成。这是第 6 章所讲述的内容。

为了更好地方便管理单元测试以及以后的测试,需要一个良好的测试管理平台,第 5 章将会介绍测试管理的原理和实践。

第5章 测试管理

从用纸笔的人工记录到专用管理软件的变迁

1946年，计算机刚刚诞生，支撑其软件的是机器语言，当时要动用大批程序员来定期维护。程序员为了使用机器语言来实现编程，需要将0、1数字编成的程序代码打在纸袋或卡片上，1打孔，0不打孔，再将程序通过纸袋机或卡片机输入计算机，进行运算。

例如完成运算 s＝768＋12288－1280，机器码如下：

```
101100000000000000000011
000001010000000000110000
```

假如将程序错写成以下这样，是很难找出错误的：

```
101100000000000000000011
000001010000000000110000
000101101000000000000101
```

为了方便调试，找出软件缺陷，需要使用笔和纸记录。随着计算机的发展，软件规模越来越大，用纸笔的人工记录逐渐过渡到字处理程序和电子数据表。

直到20世纪90年代，Rational公司（后被IBM公司收购）推出了Rational系列软件，提供了一整套软件测试管理解决方案，这样大型软件测试有了更好的选择。那就是建立在数据库或者像IBM Rational TestManager这样的商业测试管理应用软件的基础之上。

从用纸笔的人工记录到专用管理软件的变迁，反映了科学技术的发展和人们工作状态的改善。

随着软件业蓬勃发展，软件需求越来越多。在潮起潮落的IT洪流中，软件项目更是凸现大型化、复杂化的发展趋势。几十人上百人的开发团队，成千上万的模块与接口，跨地域、跨系统的使用用户等情况早已屡见不鲜。所有这些，对项目质量管理提出了更高要求，如何满足各方需求，做出更好的软件系统？测试管理逐渐成了大家目光的焦点。

软件测试管理是一种活动，它可以对各阶段的测试计划、测试案例、测试流程进行管理、跟踪、记录其结果，并将其结果反馈给系统的开发者和管理者。同时将测试人员发现的错误立刻记录下来，生成问题报告并对之进行管理，所以采用软件测试管理方法。可以为软件企业提供一个多阶段、逐步递进的实施方案。通过此管理方法，软件企业还可以用

有限的时间和成本完成软件开发，确保软件产品的质量，进一步提高计算机软件在市场上的竞争能力。

学习本章要重点掌握测试管理的内容和测试管理的流程。学会组织和控制所有测试活动、测试设计、测试自动化执行、测试评估、测试分析、缺陷追踪管理流程、缺陷追踪。还要理解测试管理引入自动化的原因，明白自动化测试是必要的，同时还能解决协同测试、分布式测试等问题。

5.1 什么是测试管理

5.1.1 测试管理的定义

软件质量一个很重要的部分就是测试和验证软件有效性的流程。测试管理是组织和控制测试工作所需的流程和工件的实践。

测试管理的整体目标是要求、跟踪、监测团队在整个软件开发工作中计划、开发、执行并评估所有的测试活动。这包括调整测试工作中包含的所有工作，跟踪测试资产中的依赖关系和相互关联，并且最重要的是对质量目标进行定义、测量和跟踪。

测试管理有助于系统地、规范地管理各种测试资源和测试活动，以提高测试的效率和质量。对于大型系统测试，测试管理工具可以帮助组织测试资产、监督项目状态、集成自动化测试工具以及度量测试效果，能够为所有这些参与者提供一个交流和协作的平台，是项目管理中必不可少的重要组成部分。

5.1.2 测试管理的基本概念

1. 测试集

测试集(test set)就是根据测试要求，对测试需求进行筛选，最终得到一组对应的测试案例的集合，称为“测试集”。执行测试集(可以手工执行和自动执行)，可帮助测试人员完成一次测试。

2. 资产注册表

资产注册表(asset registry)用作测试计划、测试用例、配置的测试用例、测试组、文件位置和迭代记录关联文件的所有位置信息的容器。

3. 配置属性

配置属性(configuration attribute)可以定义由任何配置记录使用的属性。

4. 配置值

配置值(configuration value)定义由配置属性使用的值。配置值记录与配置属性记

录相关联。

5. 配置

配置(configuration)定义受测试的系统配置。配置记录与配置的测试用例及测试组记录相关联。

6. 配置的测试用例

配置的测试用例(configured test case)记录是具有关联配置和迭代记录的测试用例的可执行格式。每个已配置的测试用例记录都与父测试用例记录相关联。单个测试用例记录可与多个已配置的测试用例记录关联,其中每个已配置的测试用例记录都将用于测试不同的配置。

7. 测试日志

测试日志(test log)显示执行的已配置测试用例记录的摘要结果。

8. 组日志

组日志(suite log)表示在关联测试组中所有执行的已配置测试用例记录的摘要结果。

9. 需求跟踪矩阵

需求跟踪是一个动态、实时的过程。这个活动的目的是为了保证用户需求与最终提交给客户的产品是一致的。通常情况下,设计成为一个矩阵模式去跟踪,纵向列出所有需求,横向表达产品开发的各个阶段。需求跟踪矩阵(requirement traceability matrix)起初是需求分析的工作产品之一,然后在总体设计、详细设计、编码、测试的每个阶段都要去跟踪,看是否全面覆盖,如果有变更则要更新本矩阵。总体设计、详细设计、编码由编码人员进行,测试阶段由测试人员进行,QA 检查过程中的每个阶段执行情况,如表 5-1 所示,在需求变更、设计变更、代码变更、测试用例变更时,需求跟踪矩阵是目前经过实践检验的进行变更波及范围影响分析的最有效的工具,如果不借助需求跟踪矩阵,则发生上述变更时,往往会遗漏某些连锁变化。借助需求跟踪矩阵,可以跟踪每个需求的状态:是否完成设计、实现和测试。

表 5-1　需求跟踪矩阵

原始需求	需求分析	设计	代码	测试
功能点 1 功能点 2 功能点 3	需求文档名称	设计文档	组件/包/类名称	测试报告名称

5.2 测试管理的内容

一般的软件开发过程包括需求、设计、开发和发布等，软件开发管理实际上就是对这几个过程的管理。软件测试过程和软件开发过程非常相似，它包括测试需求、测试计划、测试用例设计、测试执行和测试发布等过程。对这几个测试过程的有效管理，也就是软件测试的管理。

在传统的测试工作中经常会遇到这样的问题："软件需求变化太大啦，我写的测试用例都搞不清楚是验证哪个需求？我的测试用例是用 Word 和 Excel 写的，实在是太难管理！哪个阶段用哪些测试用例，我都搞不清楚了？怎么这么多构建，我不知道是谁执行了这些测试用例？这么多的缺陷，哪些是我写测试用例执行时发现的？"。要想有效地解决这些问题，必须有一套好的测试管理工具来辅助测试。

传统测试管理案例

引入测试管理工具之前，测试管理过程是这样的：根据 RUP 尽早测试的经验，从软件项目一开始，首先要面对软件需求进行测试。需求文档一般是用 Microsoft Word 编写，其中需求文档中包括业务需求、功能需求和非功能需求等几个部分。当需求文档经过评审确定第一个基线后，测试人员就可以介入。这时候工作是根据需求规格说明书开始写测试策略，进而形成测试计划概要的测试用例文档和需求跟踪矩阵文档(跟踪矩阵文档主要是将需求与测试用例对应，目的是统计需求覆盖率)。在这个过程中要重点对测试计划和需求跟踪矩阵进行管理。

接下来的设计阶段根据软件需求文档和开发设计文档，调整测试计划，进行详细的测试用例设计(包括集成测试用例和系统测试用例)。

设计阶段结束后进入编码阶段，在这个阶段会继续修改和完善测试用例文档，修改需求跟踪矩阵文档。在设计和编码阶段，重点要对测试计划、测试用例和需求跟踪矩阵文档进行管理。

在单元测试结束后，开始进入正式的测试阶段。主要过程是根据软件 Build 版本，依据软件测试计划，进行集成和系统测试用例执行，然后提交软件缺陷和编写测试执行记录。这个过程重点要对测试计划、软件测试用例、需求跟踪矩阵、测试用例执行等文档进行管理。经过多轮的测试后，系统趋于稳定，最后可以发布软件。

这个流程是一个典型的瀑布模型，这个软件开发和测试过程是按部就班的，难以应对软件开发中的不断变更。其实在整个开发和测试过程中，不变是偶然和个别，而变更才是必然和肯定。一旦不能很好地控制这些变化，文档的变更和项目的实际情况不相符，就会引发项目的危机，造成项目的混乱。因此每个阶段都强调对测试文档进行管理。

一般测试管理包括了三个方面：流程管理、资产管理和实施管理。

① 测试流程管理的目的是提供对贯穿整个生命周期的关键测试活动(如设计测试、执行测试、收集结果、分析结果)以及测试工作流(如缺陷跟踪)的支持能力。

② 测试中通常都要涉及一定的人力、设备和其他资源。测试资产管理评估的目的是评定工具对测试组织、资源分配和调度、规划安排的支持能力。

③ 测试实施管理从测试开展的角度对测试管理工具进行评估。测试实施管理评估的目的是评定工具对开展测试和实施测试的支持能力。

5.2.1 测试流程管理

测试流程管理是测试管理中的重要工作，也是测试管理工具必须具备的能力。流程管理包括以下任务：

1. 测试需求

需求定义了测试的目标，测试应根据需求来验证和确认系统。测试需求来自于系统需求，为定义测试范围、确定测试执行和缺陷的优先权、分配测试资源、规划测试任务以及分析测试覆盖提供良好的基础。

在理想情况下，测试管理工具应该提供对测试需求定义和组织的支持。并将需求集成到测试生命周期过程中的其他测试活动中，如设计、执行和发现缺陷。

2. 测试计划

测试设计建立测试项目、定义范围、分解任务、确认里程碑以及每个工作的起始和结束时间。为了支持测试设计，一个测试管理工具必须具备如下能力：

① 测试任务定义和分解；

② 为每个工作进行测试规划，定义里程碑；

③ 为每个工作分配测试资源。

3. 测试设计

可以从两个方面定义测试：测试用例和测试场景。测试用例是测试的规格说明，包括测试的前置条件、后置条件、目标需求、输入、执行步骤和期望输出等。测试场景是测试用例执行的环境说明，包括运行策略、压力设计等。测试用例和场景规格说明来自于系统需求（黑盒测试）或系统实现（白盒测试）。

通常测试管理工具需要在两个层面上支持测试定义。高一层面是通用属性的描述信息，如 ID、名称、作者等，这些一般都使用由自然语言描述的、基于模板的规格说明。低一层面是使用计算机可读的脚本，脚本能够被计算机编译或解释从而完成自动化执行。

测试定义是测试设计中的一个关键活动，设计可能是测试流程中最费时的阶段，因此。工具能够为测试定义提供有效的支持，对及时有效地实施测试至关重要。

4. 缺陷跟踪

一旦某次测试运行后没有得到预期的结果，就可能发现了一个缺陷。针对缺陷的操作包括识别、报告、评审、确认、制订优先级、修复、重新提交进行回归测试。测试的缺陷处

理过程中涉及了不同的角色，而且不同的角色之间需要相互协作。测试管理工具必须跟踪这样的协作过程以及缺陷状态的转换，特别是当测试内容更新或发生变更时，工具应该能够自动地通知相关人员，例如向开发人员通报缺陷报告，向测试人员通报缺陷修正信息等。电子邮件是及时通知方式中的一种，很多工具都支持电子邮件并能够自动触发电子邮件系统或自动发送信息。

5.2.2 测试资产管理

测试中涉及了大量的资产。除了传统的用于实施测试的计算机和网络设备外，现在的测试管理还需要考虑测试中其他“资源”，如测试脚本、测试人员等。将测试中的“硬件”（资产）和“软件”（流程）分开考虑，有利于更深入、更准确地评估测试管理工具的能力。

1. 测试资产定义

在最初的 Ad hoc 测试中，完成测试所需的硬件设备（计算机、网络设备和终端设备等）是主要的测试资产。随着测试逐步系统化和工程化，需要管理越来越多的对象，例如可以将测试设计中的很多对象（脚本、用例和场景等）当作资产的一部分系统加以管理。基于对测试的不同认识，不同的测试管理工具对测试资产有不同的定义。通常来说，测试资产划分的粒度越小、范围越广，测试管理工具对测试资产的管理能力越强。

名词解释：Ad hoc 测试

Ad hoc 原意是指“特定的，一次性的”，这里专指“随机的，自由的”测试。在软件测试中除了根据测试样例和测试说明书进行测试外，还需要进行随机测试（Ad-hoc testing），主要是根据测试者的经验对软件进行功能和性能抽查。随机测试是根据测试说明书执行样例测试的重要补充手段，是保证测试覆盖完整性的有效方式和过程。

随机测试主要是对被测软件的一些重要功能进行复测，也包括测试那些当前的测试案例没有覆盖到的部分。另外，对于软件更新和新增加的功能要重点测试。对一些特殊情况点、特殊的使用环境、并发性进行检查，尤其对以前测试发现的重大 Bug，进行再次测试，可以结合回归测试一起进行。

理论上，每一个被测软件版本都需要执行随机测试，尤其对于最后的将要发布的版本更要重视随机测试。随机测试最好由具有丰富测试经验的熟悉被测软件的测试人员进行测试。对于被测试的软件越熟悉，执行随机测试越容易。只有不断地积累测试经验，包括具体的测试执行和对缺陷跟踪记录的分析，不断总结，才能提高。

例如某测试人员拿到了一个新的 Build，按计划是进行模块 A 的功能测试，但是他灵机一动，想看看另一个功能 B 做得如何，或者想看看模块 A 在某种边界条件下会出现什么问题，于是他就来一下 Ad hoc 测试，居然在这一功能模块中发现了不少 Bug。

2. 测试资产组织

测试人员对测试的认识总是需要一个过程，随着测试过程的深入，测试将会越来越具

体、越来越细化。因此，将测试分类并分层组织是必然的，也是必要的。一个清晰的多层分组机制有助于分析测试的临界状态、分析测试的资源，还可以帮助分析测试的完备性和一致性，避免出现相互矛盾的测试和重复的测试。

测试可以根据不同的策略分组。例如可以以需求分解为一种策略，每个需求都和一组验证测试相关联，包括正常输入、不正常输入、正常路径和异常处理。另一种策略可以是系统分解，每个模块都和一组执行其各种功能和接口的测试相关。

3. 测试资产关联

测试资产关联是指测试资产，如测试用例、测试场景、测试运行和缺陷之间是相互联系的，而且也和其他软件元素相关联，如需求、软件组件等。记录这些关联信息很重要，因为一个部分的变更可能会扩散到其他很多部分，这被称作涟漪效应。一旦发生某个变更，必须定位所有受影响的部分。例如当一个功能变更请求被接受，所有相关软件模块都应被修改并确认，所有相关的测试用例都应被选出并重新确认。只有识别出所有这些依赖，并且使用文档记录下来，这个过程才能在受控的条件下进行。

4. 角色和权限管理

人员是非常特殊的一种测试资源。测试流程中通常要涉及很多部门的人员，包括设计人员、开发人员、测试人员、质量控制和项目管理人员。为了实现不同的用户可以访问不同的数据并执行不同的操作，测试管理工具必须具备角色和权限管理。基于角色的权限控制是测试管理工具中广泛使用的一种策略，设计良好的角色和权限管理机制可以为项目的安全性提供有力的保障。

5.2.3 测试实施管理

除了流程管理和资产管理，测试管理工具还需要对测试的实施提供支持。如果能够对测试的实施提供有效管理，可以显著地提高工具的易用性。

1. 工具访问

目前，软件项目趋向于更大、更复杂、更分散。测试团队通常是松散组织的，并且分布于世界各地，相互之间距离遥远。很多情况下一个团队不可能完成全部的工作，而是由多个团队协同完成。每个团队都执行子合同或者面对外包的软件模块。为了确保产品集成后的整体质量，需要所有的团队在一个统一的管理框架内工作。为了促进分布环境内的协作和合作，测试管理工具必不可少的一个功能就是，支持任何人从任何地方便利快捷地访问工具。

2. 测试执行规划和监控

可以采用两种方式执行测试：手工和自动。对于手工测试，可以使用测试管理工具记录测试执行的结果；对于自动化测试，测试管理工具需要提供更全面的支持。具体体现

在以下方面：

① 分配计算资源：为了模拟实际的使用环境，测试中通常需要涉及多台计算机。测试管理工具应该能够在分布式环境下向每台计算机分配任务。

② 规划测试运行：测试可以在不同的粒度上执行，如测试脚本、单个的测试用例/场景、测试用例组、测试场景组等。规划的目的是定义测试执行的序列，还可能需要定义在测试用例之间切换的条件，如时间限制、重复执行的频率、前置条件和后置条件等。

③ 监控执行状态：执行状态可以从两方面来说明。一方面是测试用例的状态，如激活、结束、阻塞状态下的测试用例百分比。另一方面是被测系统的状态，测试管理工具应该能够表现系统状态的各种视图，如内存利用率、CPU 使用率、响应时间、每秒钟页面访问次数和网络阻塞等。

④ 远程控制：在分布式系统测试中，远程测试越来越普遍。通常情况下测试代理器和测试控制器构成测试环境。测试代理器生成测试负载并在被测系统上运行测试，测试控制器和测试代理器之间允许交互，调度测试运行、监测状态并综合测试结果。因此，测试管理工具应该具有支持远程控制计算代理器的能力，如监控代理器的有效性和调用处理过程等。

3. 测试结果文档化和度量

为确认已经满足退出测试的条件，测试结果应被保存成文档。测试过程中记录的信息包括：

① 日期、脚本名以及测试运行的执行人。

② 测试运行的结果(成功或失败)，以及与结果相关的各种信息。

③ 任何缺陷报告的参考。

分析结果数据可以识别缺陷、修正缺陷、评估测试的覆盖效果、评估测试的有效性和生产率以及评估软件的可靠性。为了测量结果，需要使用各种度量。例如，可以使用测试覆盖度量测试的质量，常见的如路径覆盖、数据流覆盖、白盒测试的决策覆盖、需求覆盖、子系统覆盖以及黑盒测试的接口覆盖。

测试管理工具还应该支持测试执行日志和统计报告的维护，便于测试人员在需要时能够快速地找到与特定测试相关的测试日志和测试结果。

4. 测试存储库访问

测试存储库保存了所有必需的测试资产。可以从两个方面评估测试存储库的访问。首先是工具所能支持的存储类型的数量，例如文本文件、XML 文件和关系数据库。存储库支持的类型越标准、越通用、越灵活，工具的开放性就越高。其次是访问存储库的方法，集中式的访问控制能够增强安全性，而直接访问可以提供更快的响应时间。

5. 测试工具集成

自动化测试工具集成的能力也是非常重要的。测试管理工具在很高的层次上提供了项目管理和控制，而其他工具可以提供具体的、特定的自动化支持，如仿真测试和其他测

试工具集成的能力，表现了测试管理工具的开放性和扩展性。

6. 报表

报表是数据处理和分析的一种有效手段，系统必须为不同用户提供不同的视图，用及时、灵活的方式提供所需信息。因此，在整个测试流程，包括测试规划、测试设计、测试执行、测试结果分析和回归测试中，工具的报表能力是成功管理测试的一个关键因素。

定制不同的测试项目在组织结构、资产定义和流程方面有所差异。定制功能对于管理工具来说非常重要，可以帮助工具适应不同的环境。在测试管理工具中，常见的定制功能包括：

① 测试定义模板，支持用户自定义的属性；

② 工具的角色权限定义，用户权限定义；

③ 流程定义，在测试活动之上定义工作流。

7. 版本控制

考虑到缺陷修改、需求变更等原因，测试过程中需要不断地检查测试的更新情况。保持测试资产和被测试系统之间版本的一致性是非常必要的，版本控制有助于跟踪测试流程的当前状态，并为正确的软件版本维护正确的测试数据。

8. 检索

在测试流程中积累了数量庞大的数据，如前所述，这些数据被指定、引用和版本管理。测试管理工具应支持有效的检索机制，以便于用户能够便捷高效地查找所需信息。

目前市场上很多测试工具对各个测试阶段中的测试活动提供支持，目前的趋势是将各个工具组合到一起，以提供更高层次的测试支持。具有代表性的管理工具有 IBM Rational 公司的 TestManager 和 Mercury Interactive 公司的 TestDirector 等。

5.3 开展测试管理

测试管理可以分成几个不同的阶段：组织、计划、创建、执行以及报告。

5.3.1 测试组织

测试工件和资源组织是测试管理中必不可少的部分。这需要组织和维持测试项目的详细目录，以及用来执行测试的各类事物。测试组织表现了测试团队如何跟踪测试资产中的依赖和关联关系。需要管理的测试资产中最普遍的类型是：测试脚本、测试数据、测试软件、测试硬件。

5.3.2 测试计划

测试计划是回答为什么测试、测试什么、在哪里测试和什么时间测试这些问题的全部任务设置。创建一个特定测试的原因被称作一个测试激发因素，例如，确定一个特定的必要条件。为了一个项目需要，被测试的内容被分成许多测试用例。在哪里测试，通过决定和记录所需的软件和硬件配置来回答。什么时间测试通过跟踪测试的迭代或循环或时间周期来解决。

5.3.3 测试创建

测试创建是获得完成给定测试所需特定步骤的过程，它回答了如何测试的问题。例如一些测试用例被分解成更详细的测试步骤，这些步骤将变成测试脚本，要么人工生成，要么自动生成。

5.3.4 测试执行

测试用例集合建立的目标是测试执行。测试执行就是通过将测试脚本的顺序集合成测试组来运行这些测试。它回答了如何管理这些测试的问题。

5.3.5 测试报告

测试报告是指如何对测试工作的不同结果进行分析和沟通，它被用来决定项目测试的当前状态和应用软件或系统质量的整体水平。

测试工作将产生大量的信息。在这些信息里，可以提取为项目定义、度量及追踪质量目标的方法。不管使用什么沟通机制，这些质量度量方法需要被传递给其他项目，作为测试度量的基础。

测试产生的一个非常普通的数据类型是缺陷，它通常是质量度量方法的来源。缺陷不是静态的，而是随时间变化的。此外，多种缺陷总是互相关联。有效的缺陷跟踪对测试和开发团队来说十分重要。

5.3.6 测试管理中的其他因素

测试管理除了软件和硬件测试工件和资源以外，还必须管理测试团队。测试管理要调动所有团队成员的积极性，这需要对测试人员和工件进行控制、用户安全和进入许可。对于那些跨越一个或更多场所与团队的项目来说，还包括组织场所和团队协调。

良好的测试流程对测试管理十分重要。对于一个迭代式的项目，测试管理必须为测试的计划、执行和评估方法提供详尽的流程规划，与此同时，测试管理也必须提供完整的

测试策略。

5.3.7 相关的软件开发过程

虽然软件开发中所有的过程都与测试相关联，有几个与测试的关系尤为重要：

- 需求管理；
- 变更管理；
- 配置管理。

需求管理是大多数测试工作的先驱，提供了大量的测试动机和确认需求。一个项目特定的需求管理流程，对测试管理流程有重大影响。这种关系类似于一场接力赛，第一个跑的人代表着需求管理，下一个接受接力棒的人代表测试管理。

变更管理影响软件开发的全部过程，但是被跟踪的与测试工作最相关的变更是缺陷。缺陷是测试与开发之间最常见的主要通信渠道。从缺陷得出的计算和方法也经常被用作质量度量方法。

配置管理对于测试管理来说也很重要，因为在什么时候需要对哪一个要测试的版本进行跟踪，配置管理为测试执行控制着由测试管理跟踪的工作版本和环境。

5.4 传统测试管理的挑战

在软件开发中，实施传统测试管理无法较好地解决以下问题。

5.4.1 测试时间资源不足

除了某些专门的或者任务十分重要的应用程序外，很少的软件项目在开发周期里拥有充足的时间，完成高水平的质量度量。通常情况是，软件工程里本来就很短的“测试周期”总是不可避免地会被耽搁。即使是很好的项目，也极有可能在测试工作上面临时间限制。在测试管理中，这种障碍的影响就是不断变换优先级，为测试结果和质量检测方法简化数据，最后只是浅尝辄止或干脆不测了。

除了缺少时间外，有的在执行测试中取得所必需的合适资源也较为困难。例如说测试工程师、测试设备等资源可能被其他工作或项目分享，人员、设备和技术的缺乏等因素，都会对测试管理造成影响。

5.4.2 测试团队位置分散

因为测试团队位置分散，带来测试资源分布于多个区域。为了协调测试资源来避免区域之间的资源或人员不能有效利用，过去一般通过通信技术，例如电话联系、发邮件等来协调人力财力物力等问题，尽管这样，但还是存在着效率低和一些不能完全解决的问

题。例如测试管理如何协调？如何最大限度提高资源的利用率？另一区域的团队如何共享工件并保持协同合作？一个项目如何能将区域分布式开发的功效发挥到极致？这需要细致有效的协同合作，使得各地区的大多数测试人员和其他人参与到测试管理中，所以迫切地需要新技术来解决一些问题。于是可以通过开发 Web 客户端，采用网络技术，同时建立专用服务器来自动保存所有数据，并与各客户端保持同步更新。

微软是如何解决团队分散问题的？

微软亚洲工程院副院长张益肇说："我们刚刚参加完了一个与微软手机相关的项目，整个项目全期参与人员有八百多人，三大洲六个产品部门的人员一起做出来的，这是一个非常复杂的项目，如果没有很好的合作和沟通是不可想象的。"

因此微软一向很重视解决团队位置分散的问题，专门研发了很多用于团队沟通协调的工具，其中有个小工具可以在工具栏上定制全球各主要城市的时间，这样可以更好地与全球的开发团队沟通，测试工程师徐静怡说："这样，我们就不必每次和其他城市的研发中心沟通的时候，还要拿着计算器算时差了。"同时她还认为微软内部的 IT 设施非常先进，"确实微软是在提供一些很酷的技术，而且本身就在使用。"这些都有效地解决了异地协作的问题。

5.4.3 需求方面难题

需求定义了测试的目标，测试应根据需求来验证和确认系统。测试需求来自于系统需求，为定义测试范围、确定测试执行和缺陷的优先权、分配测试资源、规划测试任务以及分析测试覆盖提供良好的基础。理想情况下，测试管理工具应该提供对测试需求定义和组织的支持。并将需求集成到测试生命周期过程的其他测试活动中，如设计、执行和发现缺陷。

虽然有许多的测试策略，但是确认需求是需要完成的最主要的、优先级最高的测试工作。做到这一点需要完整的、明确的和可测试的需求。不够完善的需求管理会导致测试工作中更大的问题。

对于有效的测试管理来说，必须有对于最新系统变更和业务需求的无缝接口。这种接口不只是针对需求的描述，也要针对优先级、状态和其他属性。此外，还要开发需求说明的团队和执行测试的团队之间最大限度地协调分工和沟通。这种沟通必须在确保质量的所有方面进行。

5.4.4 与开发保持同步

测试人员与开发人员之间保持着团队协作，这是保证软件质量所必须做到的。在有些软件开发人员中间有一个错误的观点，认为关注缺陷是测试团队的事。其实对于每一个开发人员来说，了解当前的质量水平以及哪些已经被测试、哪些还没有被测试也是十分重要的。

为了使工作有条不紊地开展，测试团队必须跟上不断变化的代码、工作版本和环境。

测试管理必须精确识别要测试的工作版本和测试的合适环境。测试错误的工作版本（或功能）会导致时间的浪费，并严重地影响项目进度。测试人员必须了解什么缺陷是已知的，哪些是不需要重新测试的，以及哪些是需要确定的。然后测试人员还必须将已发现的缺陷，以及促进解决方案的信息提供给开发人员。

软件测试必须与软件开发的其他部分结合起来，特别是像需求管理和变更管理这样的规程。这包括不同的流程角色和活动之间重要协作、重要信息的沟通，以及支持这一点的集成工具。没有这些协同分工，质量将会由于缺少或误解需求、没有测试代码、没有发现缺陷和缺少关于软件质量水平的信息而降低。

5.4.5 报告正确信息

如果能够及时为项目经理报告测试状态和质量评定标准，测试工作才真正起到作用。而且要在合适的时间，为合适的人提供合适的信息，则更有助于测试工作的开展和软件项目的进展，主要有以下的原因：

- 如果只有非常少的信息，那么除了对测试团队来说减少了感知缺陷的价值外，项目涉众将不能充分了解影响质量的问题。
- 如果信息过多，那么主要信息的意义和影响就变得模糊。
- 如何将信息与不同地方的不同角色分享，总是存在着技术障碍。

报告信息的另一个需要注意的事项，是如何安排信息以及采用什么形式来展现。例如用基于工具的、基于浏览器的和基于文件的形式来展现信息。如果有技术上的限制或形式上的约束，项目涉众将得到不完全的测试和质量信息。信息应该以一种逻辑清晰的方式呈现出来，并表示适当的意义，而不是受到工具或技术的限制。因此对于项目经理来说，报告格式要考虑适应性和接受能力，这是十分重要的。

工作的价值取决于它被认知的程度，而工作如何被认知取决于传递给涉众的信息。好的测试管理必须提供完整的相关信息和正确的报告。在软件开发项目里，所有的信息，包括实时状态、目标评估方法以及结果，应该提供给所有相关的项目团队成员。

报告不应该只是传统意义上的静态文件，因为项目的信息是持续变化的，为了准确地交流信息，需要能实时输出信息的多种形式。随着项目的进展，这些都会帮助项目的不同角色对信息变化做出正确的反应。

来自不同规程的信息不是完全独立的，因此来自测试管理的输出，可以与其他规程的信息结合起来。当前的技术，使得将所有的规程信息结合成统一视图成为可能。这样可以确定所有规程的健康状态，而且也使得清楚地展示评估测试、开发和其他项目工件之间的关系成为可能。

5.4.6 测试管理的评估

测试管理也需要评估，这样才能知道测试管理是否混乱或者严谨。评估决定质量，是测试团队的一个主要目标，但是如何准确地度量质量呢？有许多方法可以实现，而且根据

系统或应用软件的类型和开发项目的特殊性，可分为很多不同的种类。为了避免曲解，任何一个质量度量方法都是需要清晰明确的。更重要的是，测试方法必须可以获取和保存，否则可能是不完整或不准确的，也将不值得去实施。

为项目确定质量目标，并决定如何有效而准确地测量这些目标。测试管理是详细说明目标、用于测量这些目标的方法，以及如何收集这些数据的地方。测试中许多工作可能没有明显的完成标准。定义正在进行的流程、变更特定的输出和测量方法，将更详细地说明测试工作的活动和任务。牢记测试的特定目标和测试方法，不仅有助于跟踪状态和结果，还能避免最终将所需报告混在一起。

在单一公共的知识库或数据库存储测试管理的结果，可以确保对它们进行分析或使用，并有效地促进工件（包括工作）的版本控制，避免出现过时或无效信息的问题。这一切将有助于项目成员了解流程，并在测试工作的基础上做出决策。

5.5　基于 RUP 的测试管理经验

为了解决传统测试管理所遇到的挑战，RUP 总结了软件测试管理的经验，主要是以下四点。

5.5.1　尽早开展测试管理活动

目前，在实际工作中，能尽早开展测试管理活动的软件项目还不太多，相信它的好处将会逐渐被人们所认识。尽管在项目早期，确定测试资源很难，但许多测试分析如识别关键的优先测试用例，可以且应该尽快开始。一旦用例被充分开发产生事件流，就可以得到测试程序。如果一个项目没有使用用例需求，那么仍可以从确认初始需求说明中得到测试。尽早开展测试和测试管理活动，能帮助减轻未来不可避免的工期紧迫所造成的压力。

5.5.2　迭代化测试

软件测试是一个反复的过程，在整个项目周期的早期，生成有价值的测试工件。RUP 建议一个迭代的测试流程应很早就关注测试管理。为了达到这个目标，测试管理可以组织迭代的各类工件和资源。这个基于风险的方法，有助于确保有效处理项目可能出现的变更、延迟和其他一些不可预见的障碍。

5.5.3　重用测试工件

在一个项目或多个项目里，重用测试工件，能够极大地提高测试团队的工作效率，缓解工期和资源有限造成的压力。可以重用的工件，不仅包括自动操作测试对象，还包括测试程序和其他的计划信息。为了有效地重用工件，测试管理必须很好地组织和描述给定

项目的与测试相关的各种信息。在创建工件时，重用需要预先计划。这个原则通常可以应用于测试管理。

5.5.4 定义执行灵活的测试流程

一个好的、可重复的流程有助于测试人员了解项目当前的工作状态，并能够根据经验预测并了解下一步的工作。但是，各个项目都有自己的特点，测试工作的目标和需要使用的技术也不尽相同，因此测试管理的工作流程，除了定义明确且能够重复执行，还应该是可以根据需求来进行修改的。设计流程时应该考虑到这点，如果对原有流程进行修改十分容易操作，这样在迭代项目过程中，便可通过调整流程来提高工作效率。

5.6 测试管理的自动化

软件测试管理自动化是实现软件测试自动化的基础。测试管理的内容有很多，而且许多工作非常耗时。为了节约时间，可以使用工具让许多工作自动化或半自动化。虽然像字处理程序和电子数据表这样的简单工具已经提供了很大的灵活性，但专门用于测试的自动化工具更加有效，更加有助于节约时间。

5.6.1 引入测试管理自动化的原因

传统测试管理流程存在着大量的弊端，一个变更就可以造成很大的影响，这也是引入测试管理自动化的原因。下面来看看是如何变更的。

当需求确定初始基线后，需求的变更就正式开始了。在开发设计初期，需求变更就会频繁到来。当进入需求开发阶段，对需求的新增、修改、删除和延期状态的变更被审批通过后，要做的就是在一个约定的阶段点，调整测试计划中的测试范围，随之测试任务也会发生变化，这时需要调整测试任务的进度和安排，根据任务的变化来调整人员的工作量，还要增加和修改概要测试用例，增加、删除、修改需求跟踪矩阵文档，以免有些变更的需求被遗漏。这期间一般是通过版本控制软件来管理文档，人为地在文档中加入修改信息，标识出修订人、修订时间和主要修订内容。

当总体设计和详细设计完成，进入编码实现阶段，如果前期需求分析做的不是很到位，这时变更需求的工作量会加大，一般表现在新增、修改的需求会比较多。需求基线会依据项目情况进行变化、发布，测试人员根据这些基线，开始修改测试计划，新增、修改和删除一些测试用例，相关人员更新需求跟踪矩阵文档。

等到编码阶段结束后，进入集成测试和系统测试阶段时，大量的软件缺陷被发现，反馈给开发人员后，有些缺陷会引发需求变更，同时加上日构建的影响，使测试人员疲于奔命。大量的手工测试任务开始全面实施，尤其在测试人力不足的时候，测试人员在这个阶段所做的工作会受到很大的影响，只好从按照测试计划执行，到随意调整测试计划；从按

照测试用例执行，到放弃测试用例，并进行大面积的随机测试和回归测试。在这个阶段，如果规划得不好，还会出现一些严重问题，例如软件需求变化过于频繁，难以确切了解发生的变更，这些很可能对测试造成很大影响。一旦需求变化，哪些测试用例需要修改？测试用例是验证哪个需求的？各个阶段需要用哪些测试用例？构建版本是不是太多？这些问题的出现，使规范软件测试过程的核心测试文档已经不起作用了，只好重新回到了无计划、无规范、无文档的测试状态。最后终于在大量的加班和项目延期的情况下，在项目组成员满腹牢骚的状态下，软件产品没经完全测试的状况下发布了。

什么是日构建

传统开发软件的流程一般是这样，理解领域问题，然后分配任务，由不同的人负责不同的软件部件，在开发完成之后，再把各人的部件整合起来，形成完整的软件。这个思路看起来并没有什么问题，但是在实践中却问题很多。

首先，这种方式适合开发人员之间工作彼此没有交集的情况，以前这种现象很常见，但是现在，随着软件规模的扩大、分工合作的加深，开发人员间的相互依赖程度越来越高，这种清晰的职责划分已经变得越来越难了。

其次，在软件集成时，往往会出现各种各样的问题，可是却很难发现到底问题在哪里？每个人的代码都没有问题，结合到一起就出现大量的问题。

所以日构建就将平时难得一见的集成工作转换成频繁进行的一件工作，从而使得原先如同噩梦般的集成变成了一件简单的工作。这也是很容易理解的，如果集成工作几个月才进行一次，谁能够记起几个月前的细节呢？但是如果集成以天，甚至以分钟为单位进行，排除 Bug 就变成一件很容易的事情了。

我们看到，当需求变更比较频繁时，原本比较有序的开发、测试过程被打乱了。为什么会出现这样的结果？

Word 格式的需求文档，在需求变更频繁时很难准确标识出每个变更之间的区别，换句话说，相关人员也就很难准确得到需求变更的信息。这会为项目组成员之间的沟通造成一些障碍。

对于测试用例设计工作就更麻烦了。在得到需求变更信息后，测试人员增加、删除和修改 Word 格式的测试用例，使其适合当前的需求版本，接着再去维护需求跟踪矩阵的状态标识，确认哪些需求得到了测试，哪些需求已经编写了测试用例等。由于需求跟踪矩阵是一个二维表，本身管理起来就很烦琐，还要进行多次的新增、删除和修改的操作，维护量非常大，在这个阶段常常会听到大量的抱怨。

在运行测试用例时，测试人员需要填写测试记录，例如，本次执行是在哪个 Build 下完成的？执行了哪些测试用例？执行的结果是通过还是失败？由于测试用例数量繁多，其本身的维护量也是很大的。这个过程全部都是手工测试，测试人力不足在后期的表现非常明显。测试人员也会人为地简化测试步骤，减少过程记录或者编造记录，造成后期的度量数据不真实，过程改进工作也会受到影响。

整个过程中，发现需求文档变更，测试计划就得调整，需求跟踪矩阵文档就必须做相应的修改，测试用例文档也必须更新，测试执行也会根据变更的测试用例进行相应的调

整。这是一串连锁反应，而 Word 格式的测试文档是一个个独立的相互之间没有联系的文档，不太可能在文档之间建立某种关系来自动做出响应，但是它们之间又必须存在关联关系才可以解决问题。依靠手工来维护这种关联关系，往往只能达到事倍功半的效果，所以解决问题的关键，在于有效地解决“关系”问题，达到事半功倍的效果，在技术上解放自己。

5.6.2 测试管理自动化

测试管理工具很多，这里以 IBM Rational TestManager（以下简称 TM）工具为例，介绍测试管理工具是如何做到软件测试管理自动化的。

TM 是 IBM Rational 公司根据 RUP 统一软件开发过程开发的一套自动测试管理工具，它是一个开放的、可扩展的框架，它将所有的测试工具、工件和数据组合在一起，主要对被测项目从需求到测试计划，从测试计划到测试用例设计，从变更、缺陷跟踪到测试结果分析，全程进行管理和控制，从而提高团队的工作效率，并可以帮助开发团队加速应用程序的开发和测试团队的测试和实施。

从图 5-1 可以看出，在整个过程中，TM 通过关联关系，将需求、测试计划、测试用例设计、测试执行和测试评估等过程进行整合，非常符合软件测试管理自动化模型。这些关联关系是用 TM 数据库中的记录来表示的，而数据库中的记录是按照 TM 定义的层次结构来组织的。

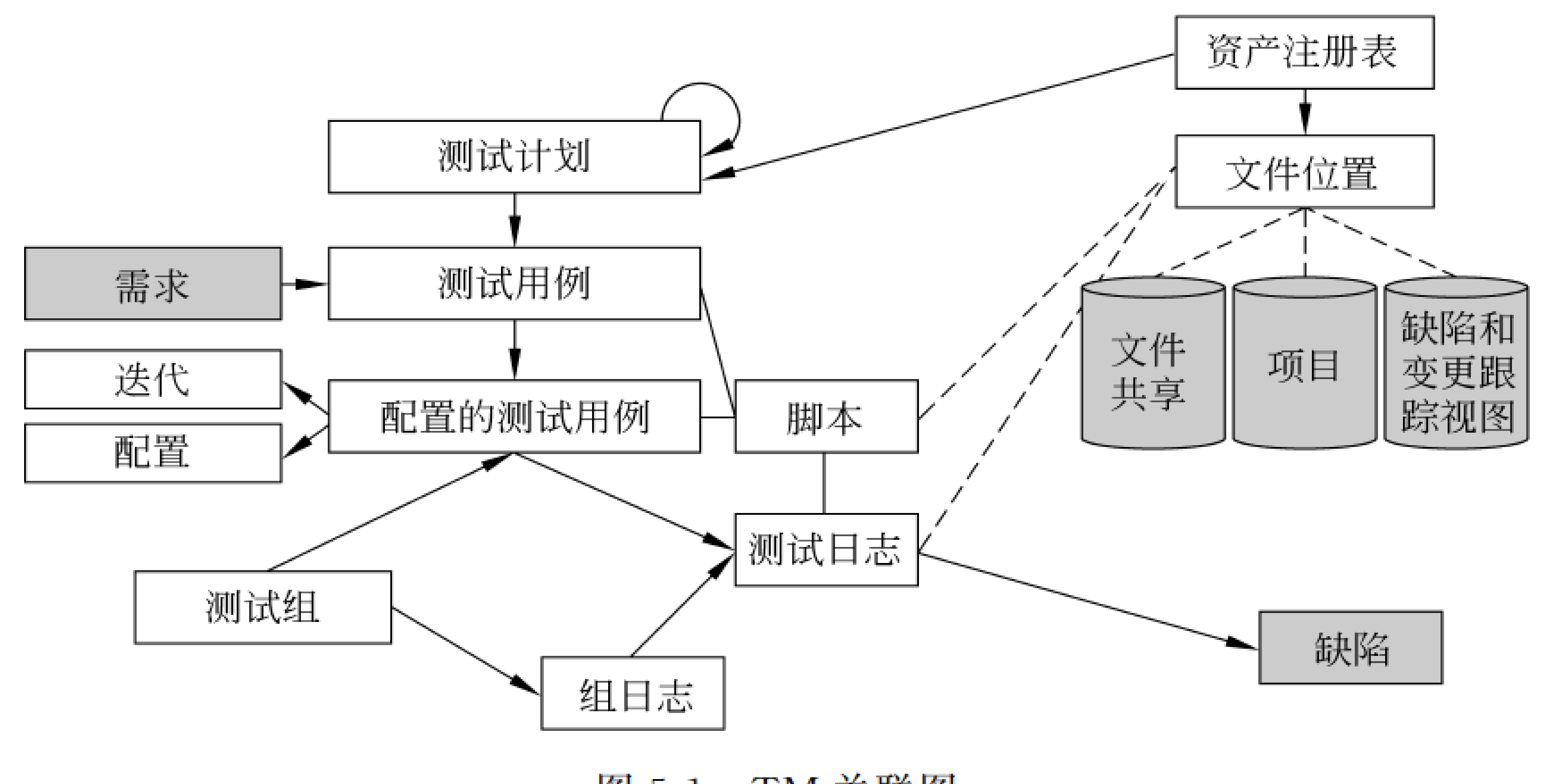


图 5-1　TM 关联图

目前市场上测试管理工具都引入关系型数据库来解决关联的问题，例如将上述的 Word 文档内容存入数据库各个表中，通过表间的某种关系，实现各个表中的数据自动关联，就可以有效地解决人为处理文档关系不当造成的影响。所以提出一个概念“关系型测试管理”。通过这个概念来设计一个“软件测试管理自动化”模型。模型可以分成三个部分：

第一个部分是需求与测试计划的关联。它们之间的关系表现为多对一或者多对多，对于功能测试，可以定为多对一的关系。当新增、修改和删除某些需求时，相对应的测试

计划自动标识为怀疑，测试人员发现后会通过对应关系找出有哪些需求发生变化，然后调整测试计划。

第二个部分是需求与测试用例的关联，它们之间的关系表现为多对多。当新增、修改和删除某些需求时，相对应的测试用例自动标识为怀疑，测试人员这时可根据变更的需求修改相应的测试用例。

第三个部分是测试用例与测试执行的关联，它们之间的关系表现为一对多。当新增、修改和删除某些测试用例后，会自动标识此测试用例已经变更，需要测试。这样测试人员会根据变更标识去执行更新后的测试用例。

有了上述的关联关系，相应的需求跟踪矩阵、需求覆盖率统计、测试执行文档就可以自动生成。而且也可以自动生成每次需求基线变化后的测试计划文档和测试用例文档。这样可以时刻地关注测试过程中的变化，也就可以解决上述的一些问题。人工要做的就是对软件测试过程和结果进行评估和分析，更大的发挥人的聪明才智。

在需求阶段，当需求基线发布后。为了在 TM 中对需求与测试计划、测试用例和测试执行建立关系，需求是一个最常用的做测试计划的信息来源。使用需求作为测试开发的基础，只需要把这些需求保存在 Rational RequisitePro（IBM 需求管理工具）或 Microsoft Excel 中。所有这些来源都将被输入到 Rational TM。把测试连接到需求的好处，是能够验证每个需求的测试以及可以对照需求列表报告测试进展。尽管创建的很多测试用例都不在测试需求的范围内，但是使用需求指导测试计划仍然是一个好的开始。

将需求作为测试输入导入到 TM 中，一般是创建资产注册表，再创建测试计划、测试用例、配置的测试用例等记录类型，用以形成测试层次结构。然后根据测试计划，在 TM 中设计测试用例，这时要进行概要的测试用例设计，这也体现了 RUP 中迭代的思想。设计测试用例时，重点要将测试输入与相应的测试用例进行关联。

概要测试用例设计结束后，需要在开发的设计和编码阶段，进行详细测试用例设计。这项工作体现在测试用例的实现中，它可以将一个概要的测试用例细化成详细的手工测试脚本和自动化测试脚本。加入了自动化测试，可以有效地解决手工测试执行周期长的问题。该阶段的工作是创建测试脚本，并与测试用例和配置的测试用例相关联，进行对整个测试的管理，尤其是对测试脚本、测试结果进行管理。

再来看看在需求变更下，TM 是如何做到自动化测试管理的。首先产生变更需求后，需求管理人员修改相应的软件需求。由于测试输入并不是一种单纯的导入，它依然和原来的需求保持了一一对应的关系，所以当原需求发生变化，测试输入也需要做相应变化。

对新增状态的需求，在 TM 中并没有测试用例相对应，但是 TM 可以通过需求与测试用例的关联关系，很快发现新增需求没有进行测试用例设计，如图 5-2 所示。

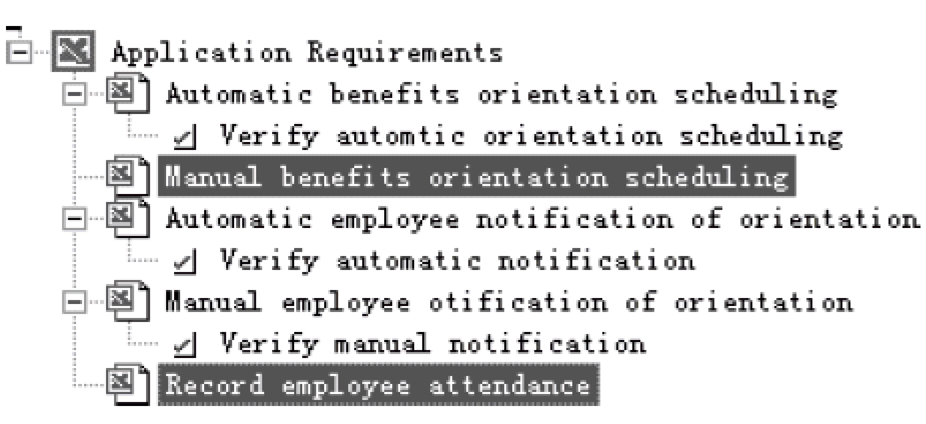


图 5-2　需求与测试用例关联

对修改状态的需求，由于在 TM 中有相应的测试用例与之对应，所以立刻会对相应的测

试用例标识为怀疑，等待测试人员进行相应修改。

对于需求的任何变化，TM 都可以对需求进行自动化跟踪，完全可以代替需要手工去更新的 Word 格式或 Excel 格式的需求跟踪矩阵文档。

最后执行配置的测试用例或测试组，评审测试结果，如果结果适当则提交到 TM 数据库中形成测试日志或测试组日志，完成对测试的执行，并配合记录结果，再进行结果分析。TM 就是利用需求、测试用例和测试执行的关联关系，有效解决需求变更带来的问题，同时也解决了烦琐的手工测试管理带来的问题。并且在 TM 中，通过细化的手工测试脚本和自动化测试脚本的结合，大大地提高了软件测试的效率。通过自动生成的测试报告和测试文档，可以对测试工作进行有效评估，从而更有效地保证软件质量，所以软件测试管理自动化是可行的，而且也是必要的。

还可以看出测试管理平台有以下优点：

第一，协同测试。测试是一个团队，不是一个人单独做，因为一个人只能测一点，每个人之间要有分工，分工要有合作，整个项目是一个交集、合集，怎么了解分配状况，需要集中统一的平台，所有数据在平台上交流，所有人可以看到相应的数据，这样数据可以被公开和跟踪。

第二，分布式测试。例如有 200 个测试，要分布在 10 个机器上，测试分布怎么做，最笨的办法就是一个一个系统去做，如果远程做，代价高，平台就可以解决这个问题。

第三，能适应测试要求。例如设计银行信用卡，假设设计 5～10 年以后有 1 亿用户，但现在可能只有一两百个用户，怎样保证系统设施满足将来的需求，只能通过测试方法，平台就可以提供并发的测试环境，而且可在里面输入不同的测试数据、不同的测试流程，这是自动化测试的好处。

5.7 TM 的使用

TM 可以进行测试计划、需求覆盖、测试用例、压力测试、测试执行和测试评估等方面的管理，这样就可以用这些功能来帮助管理软件测试工作。

下面通过测试流程、测试输入、测试计划、测试用例设计、测试实现、测试执行和测试评估等 8 个部分，以图文并茂的方式一一说明。

从图 5-3 中可以看到 TM 可以创建和运行测试计划、测试套和测试脚本，可以插入测试用例目录和测试用例，进行测试用例设计，对迭代阶段、环境配置和测试输入进行有效的关联。可以创建和打开测试报告，其中有测试用例执行报告、性能测试报告，以及其他很多报告。

除此以外，TM 还有很多辅助的设置，其中包括：创建和编辑构造版本、迭代阶段、计算机、计算机列表、配置、配置属性、数据池、数据类型、测试输入类型、测试脚本类型等，还可以定制系统需要的属性。

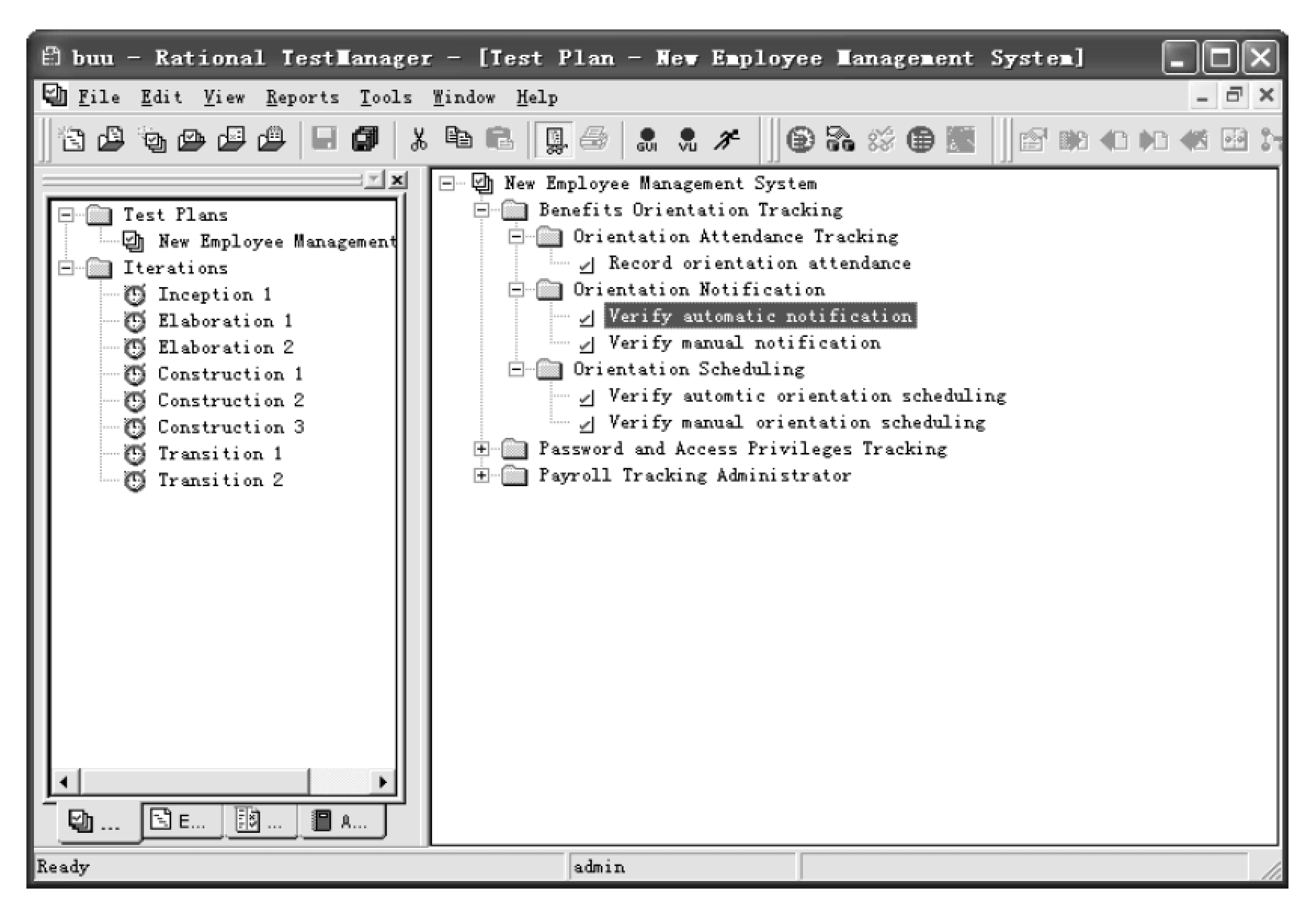


图 5-3　TM 界面

5.7.1　测试流程

下面来看一个最简单的和常用的测试流程：根据软件需求规格说明（Word 格式）文档，书写（Word 格式）测试计划，设计（Word 或 Excel 格式）测试用例，根据软件 Build 的版本执行测试用例，记录（Word 或 Excel 格式）执行结果，发现并提交（Word 或 Excel 格式）缺陷报告给开发人员，跟踪所有缺陷直至解决，提交最终（Word 格式）测试分析报告。

从图 5-4 中可以清楚地看到，IBM Rational 将所有的需求作为测试输入，根据测试输入来制定测试计划，整个测试计划的核心就是规划、组织和设计测试用例，通过手工测试和自动化测试两种方式来实现测试用例，然后进行测试用例执行，记录测试用例执行结果，并将发现的缺陷提交到缺陷管理系统中，最后对测试结果进行评估。

TM 将需求、测试计划、测试用例设计、测试执行、测试报告和测试缺陷全部集成，通过相互的关联关系，更体现了其强大的自动化管理的功能。

5.7.2　测试输入

计划测试工作的第一步就是验证测试的输入。测试输入就是测什么问题，以及哪些需要验证。测试输入帮助决定什么需要测试，而且帮助确定当开发过程的基线发生变化时，那些测试需要改变。在迭代开发过程中，由于各种变更非常频繁，所以这点非常重要。

TM 内置了三种测试输入类型，即 IBM Rational RequisitePro 中的需求、IBM Rational Rose 中的可视化模型和 Microsoft Excel 电子表格中的数据。其主要目的是将输入的测试需求与相应的测试用例、测试执行关联，便于对需求进行跟踪确认。

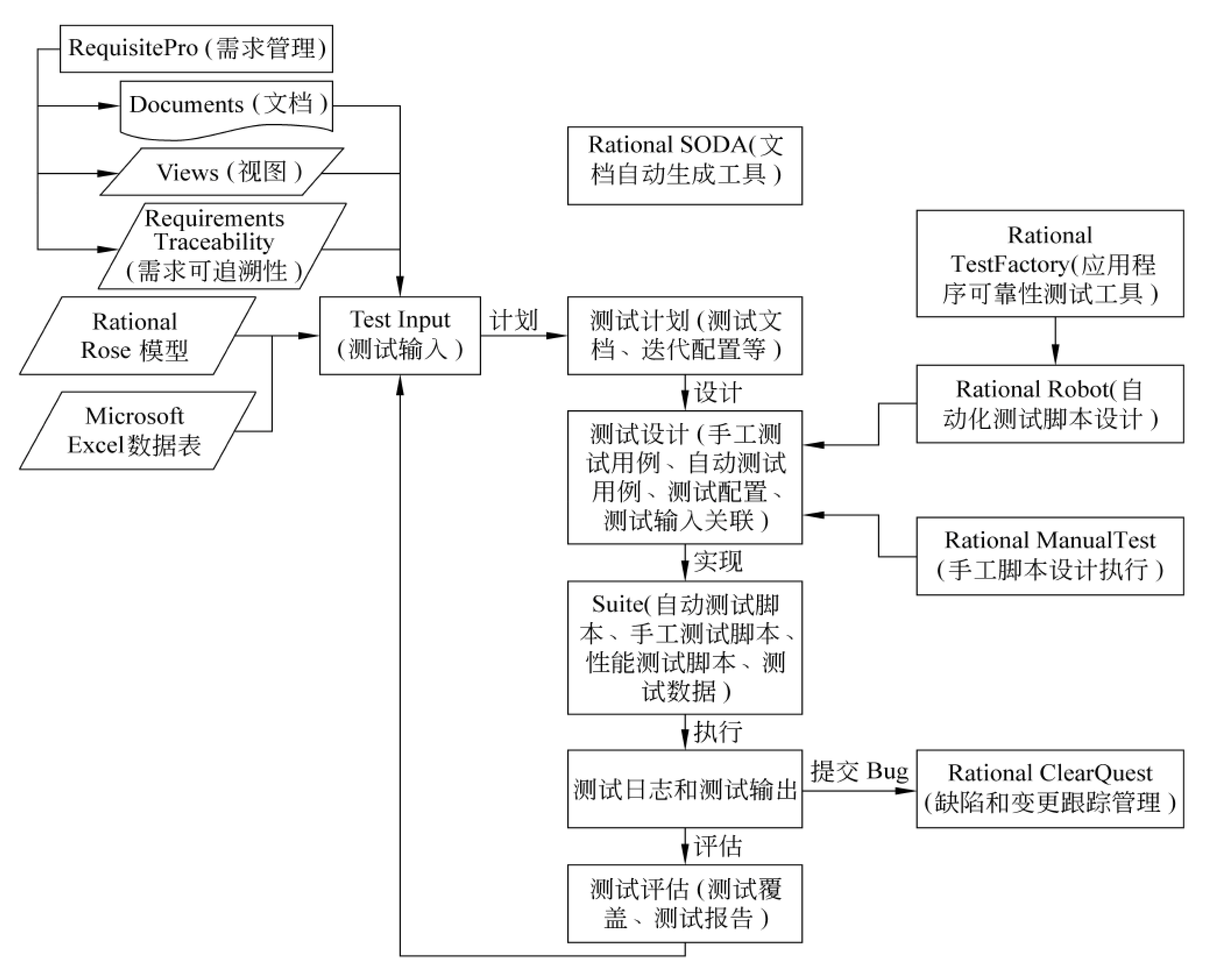


图 5-4　TM 流程图

测试输入的实例是某员工管理系统的一份 Excel 的需求文档,如图 5-5 所示。

Microsoft Excel - TMTutorial_Requirements

B1　Requirment Description

	A	B	C	D
1	Requirement Name	Requirment Description	Status	Last Modified
2	Automatic benefits orientation scheduling	The application will automatically prompt the administrator to schedule the new hire for the next available benefits orientation.	Approved	8/20/08 12:00 PM
3	Manual benefits orientation scheduling	The application will allow the administrator to manual schedule the benefits orientation.	Approved	8/20/08 12:00 PM
4	Automatic employee notification of orientation	The application will send an email to each new employee containing the date,time and location of their benefits orientation.	Approved	8/20/08 12:00 PM
5	Manual employee otification of orientation	The application will allow the administrator to send an email to each new employee containing the date,time and location of their benefits orientation.	Approved	8/20/08 12:00 PM
6	Record employee attendance	The application will allow the administrator to record the date of the new hire's attendance along with any questions that the employee may have.	Approved	8/20/08 12:00 PM

Sheet1 / Sheet2 / Sheet3

图 5-5　Excel 格式需求文档

通过 TM 中的测试输入的属性将 Excel 格式的需求导入到 TM 中，通过选择将 RequisitePro 中的需求关联，这样测试输入就设置好了，可以开始制定测试计划，详细如图 5-6 所示。

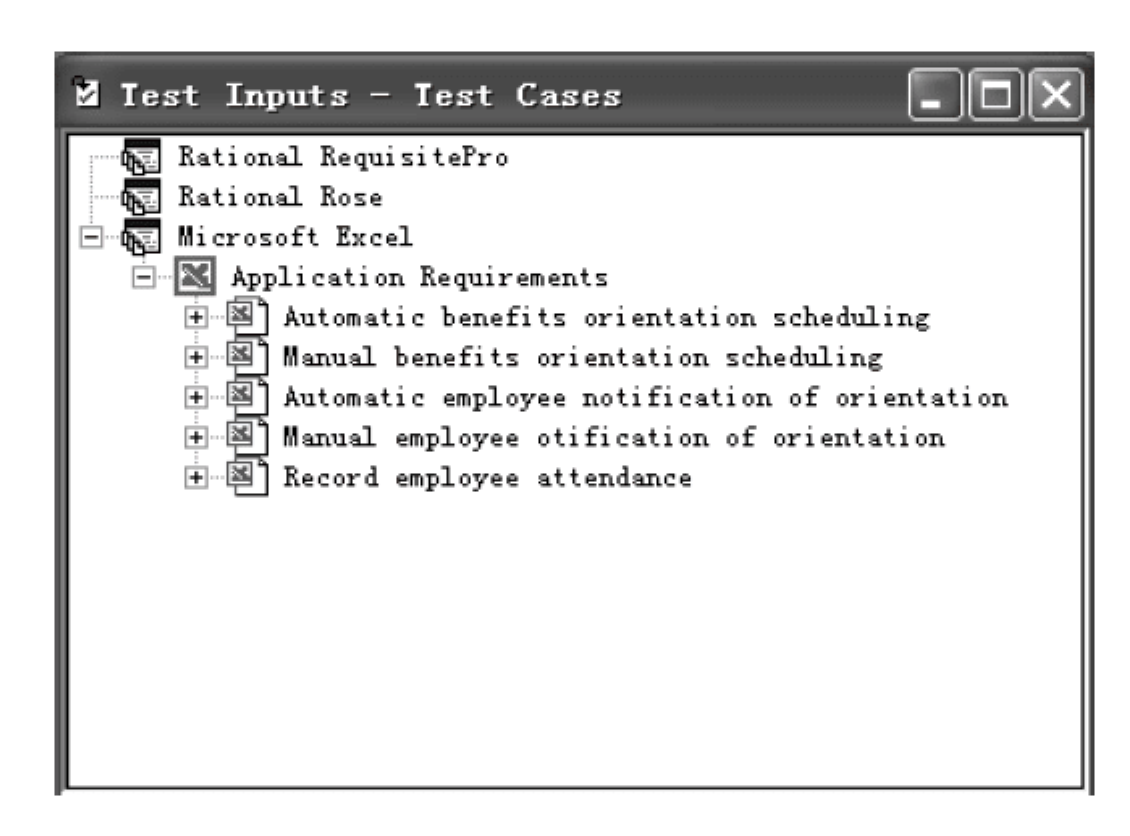


图 5-6　需求列表

5.7.3　测试计划

书写测试计划要解决以下几个问题：第一个是 What 和 Where。主要是通过需求、可见的模型和其他测试输入来告诉测试员哪些需要测试，在哪里测试；第二个是 Why。测试输入可以告诉为什么要在某些地方进行测试，即通过需求的优先级逐级地对系统需求进行验证；第三个是 When。迭代计划可以告诉什么时候进行测试，什么时候必须通过测试。第四个是 Who。测试计划、迭代计划或项目计划告诉谁去执行这些测试活动。

那么，明确了这些，而且系统功能需求已经作为测试输入导入到了 TM 中，这样就可以开始制订测试计划。

在日常的测试工作中，测试计划文档通常都是比较详细的，RUP 测试计划模板中一般会包括：简介(目的，背景，范围，项目标识)、测试需求、测试策略、测试类型(数据和数据库完整性测试，功能测试，业务周期测试，用户界面测试，性能评价，负载测试，强度测试，容量测试，安全性和访问控制测试，故障转移和恢复测试，配置测试，安装测试)、工具、资源(角色，系统)、项目里程碑、可交付工件(测试模型，测试日志，缺陷报告)、附录 A(项目任务)等。

那么 TM 中的测试计划是否就是传统的测试计划文档呢？其实不然。传统的测试计划在这里可以作为一个项目的总体测试计划，而 TM 中的测试计划主要是将总体测试计划细化，针对系统功能和性能部分，进行规划、组织和详细设计测试用例，其中包含几个主要的任务：收集和标识测试输入(需求)、建立测试计划、创建测试用例文件夹、创建测试用例、定义测试配置、定义迭代阶段等。在 TM 中，可以外部关联整体测试计划文档。其实 TM 最擅长是对系统的功能和性能测试的管理，所以通常将这部分计划用 TM 来管理和跟踪，而测试进度这部分可以由类似于微软的 Project 工具来管理和跟踪。

下面通过实例说明在 TM 中如何制定测试计划。首先新增加一个测试计划，计划名称为 Benefits Orientation Tracking，具体操作：单击菜单 File→New Test Plan，接着创建两个测试用例文件夹 Password and Access Privileges Tracking 和 Payroll Tracking Administrator，最后将总体测试计划外部关联。在 TM 中的具体表现形式如图 5-7 所示。

测试计划制定完成后，接下来要进行测试用例设计。

5.7.4 测试用例设计

测试设计的目的是回答如何去执行测试这个问题。测试设计主要是根据测试计划进行功能和性能测试等方面的设计，对于系统测试过程，主要包括测试条件、前置条件、测试步骤、验证点、后置条件和测试通过标准等几个方面。良好的测试设计是测试自动化的重要保证。

在这里还是通过实例开始进行测试用例设计。首先将 Password and Access Privileges Tracking 和 Payroll Tracking Administrator 测试用例文件夹细化分解成几个测试用例。

具体设计方法：创建测试用例，选中测试用例文件夹，单击菜单 Edit→Insert Test Case(见图 5-8)；在 New Test Case 对话框中，在 General 标签单击 Design 按钮，在 Design Editor 窗口中，分步描述概要的测试用例(见图 5-9)；在 Test Inputs 中加入与测试用例相对应的测试需求，这样就将测试用例与测试需求关联起来，对于产生变更的需求，系统会自动将其设置成怀疑(Mark Suspect)的标记，这样就知道哪些测试用例需要修改；在 Implementation 标签中，可以设置此测试用例的实现方式，是手工还是自动化，并需要详细说明测试完成的前提条件、置后条件和通过标准。



图 5-7 测试计划外部关联

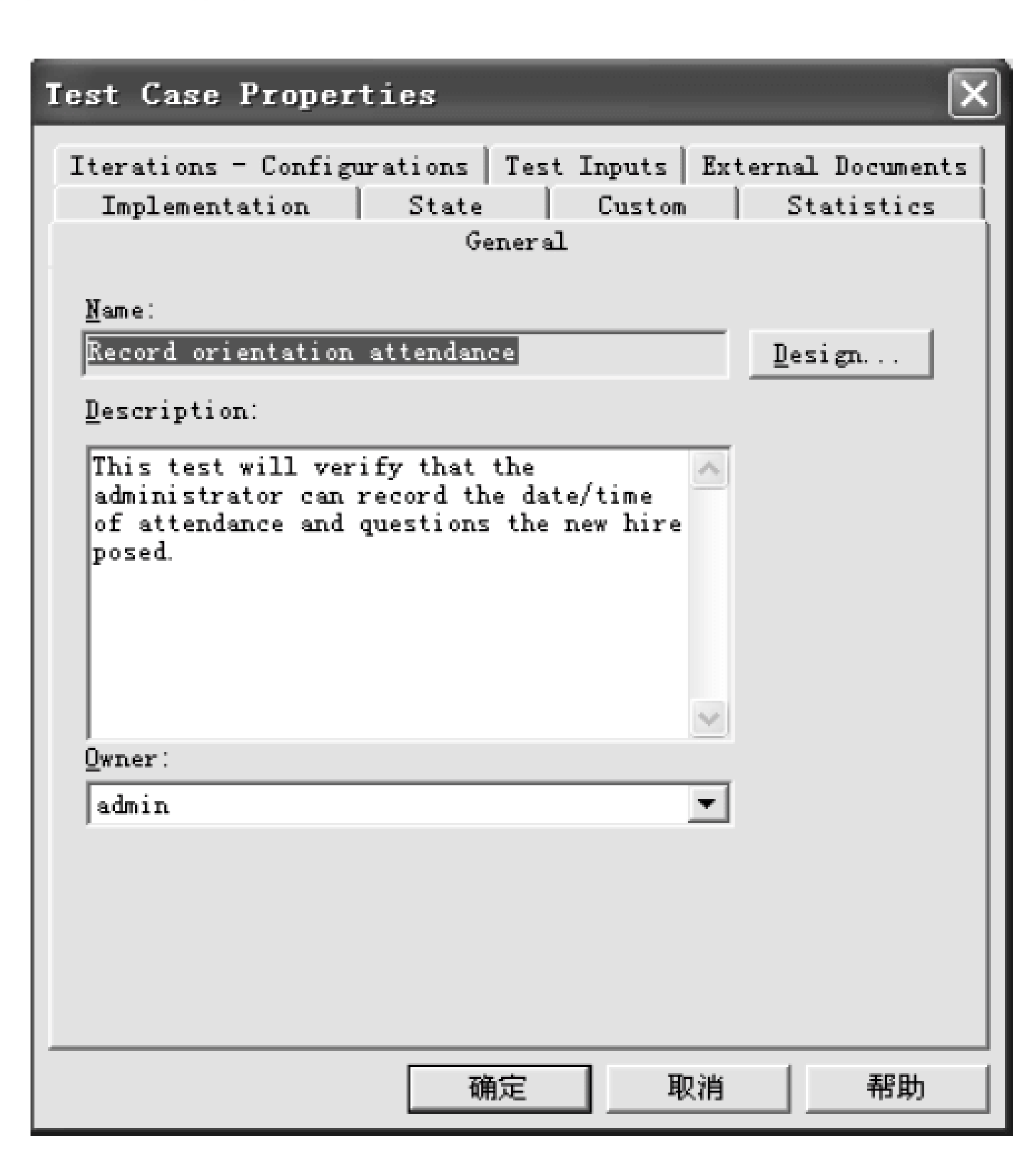


图 5-8 测试案例属性框



图 5-9　设计概要测试用例

注意：TM 中在 Test Case Design 和 ManualTest 设计中，用惊叹号来表示 Step 即步骤，用蓝色的对勾表示 VP 即验证点。

5.7.5　测试实现

在日常的测试工作中，基本上没有测试实现这个概念，在 TM 中增加的这个环节，有比较突出的好处。测试实现主要有几个部分组成：在测试脚本中调用测试脚本服务、创建手工测试脚本、测试用例实现关联和在 Suites 中实现测试套等。TM 中内建两种实现方式，一种是手工实现，对应的是手工测试脚本；另一种是自动化实现，对应的是自动化测试脚本，一般是采用 IBM Rational Robot 测试脚本。

自动化测试脚本必须在 Robot 中录制实现，一种是 GUI 脚本（菜单 File→New Test Script→GUI），是用 SQA Basic 编写的，主要用于功能测试；另外一种是 VU 或 VB 脚本（菜单 File→New→Test Script VU），主要应用在性能测试，录制一个 Session，如图 5-10 所示。

手工测试脚本创建一般有三种方法：第一种是从文本中导入，第二种是从测试用例 Design Editor 中直接输入，第三种是在 IBM Rational ManualTest 中实现的。手工测试脚本主要由详细步骤、预期结果和验证点组成。

TM 中还可以通过创建 Suites 实现，其中 Suites 中可以包含测试脚本、测试用例和其他项，一般创建 Suite，主要的目的是将相互关联的测试用例集成在一起执行。为了建立一个新的 Suite，需要单击菜单 File→New Suite，然后通过 New Suite 向导，一步一步进行，直至最后完成。除此以外 TM 还可以定制测试脚本类型，通过操作菜单 Tools→Manage→Test Script Types，再单击菜单 New 可以建立。

以上主要介绍了测试用例的实现方式，那么在所有测试用例都设计好以后，接下来要做的事情就是将 Password and Access Privileges Tracking 和 Payroll Tracking Administrator 测试用例文件夹中的几个测试用例逐一进行测试实现。

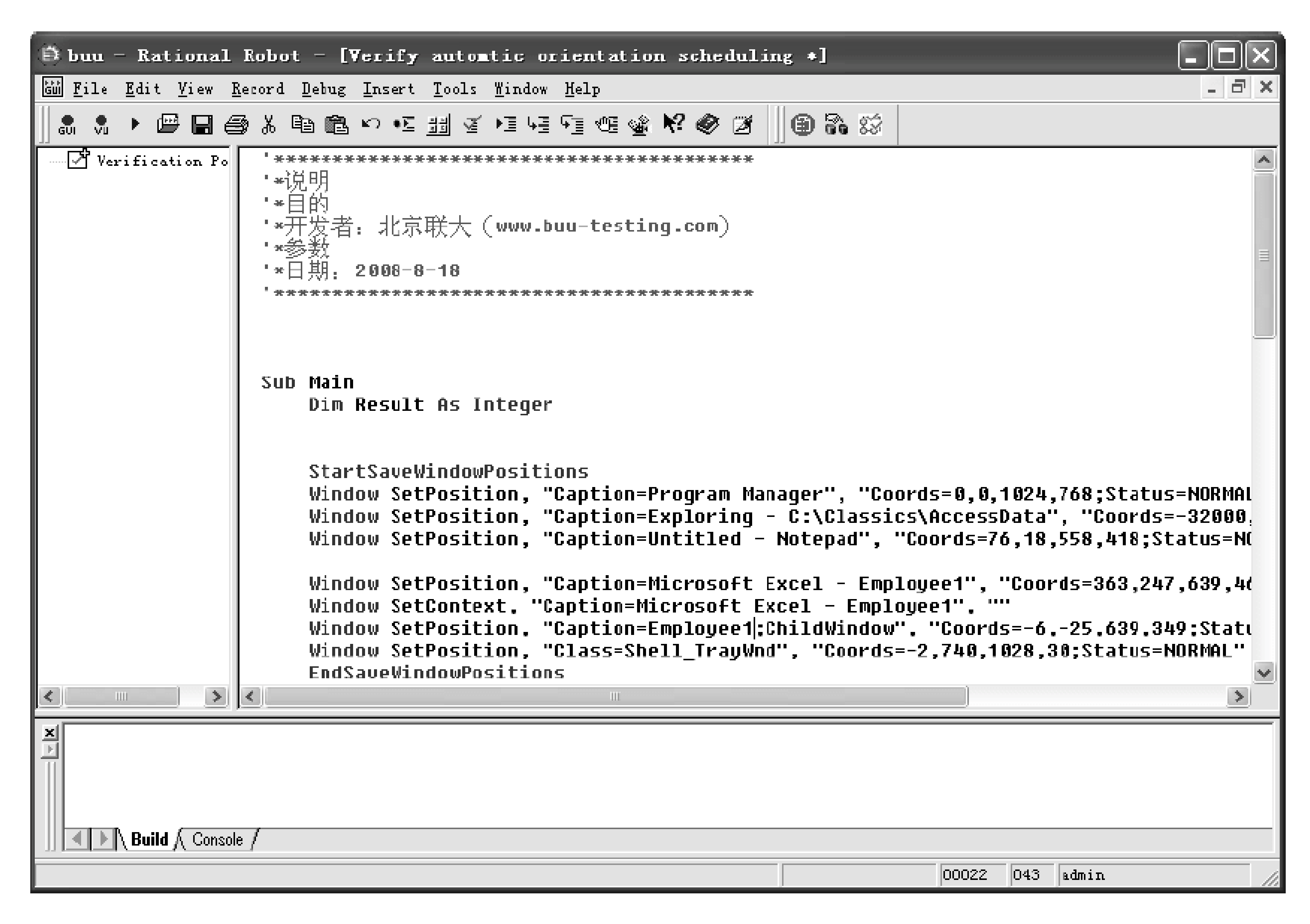


图 5-10　编写 Robot 脚本

5.7.6　测试执行

测试执行主要是进行测试执行管理、进行测试、记录测试结果，包括缺陷报告和测试日志。具体包括几个部分，分别是执行自动测试脚本、执行手工测试脚本、执行测试用例、执行 Suites、监视测试运行和测试停止。

在测试执行时，首先要设置好软件 Build 的版本，设置好运行的机器，即操作系统配置，然后根据测试计划，分别开始执行测试用例。具体操作如下：

运行自动测试脚本，单击菜单 View→Test Scripts，选择一个 GUI 脚本，单击鼠标右键，在弹出的菜单中单击 Run ，或单击菜单 File→Run Test Script，选择一个脚本运行。自动测试脚本运行后，会自动执行自动测试脚本，如图 5-11 所示。

运行手工测试脚本，单击菜单 File→Run Test Script→Manual，选择一个脚本，或单击菜单 View→Test Scripts，选择一个 Manual 脚本，单击鼠标右键，在弹出的菜单中单击 Run，如图 5-12 所示。

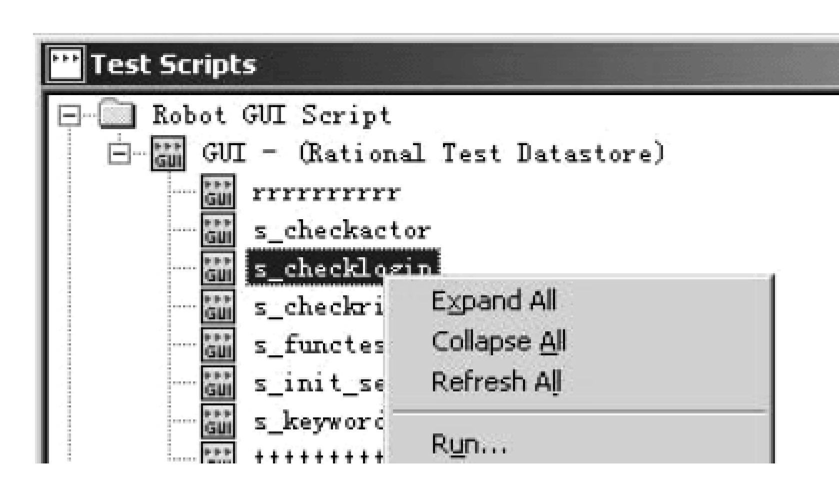


图 5-11　选择脚本执行

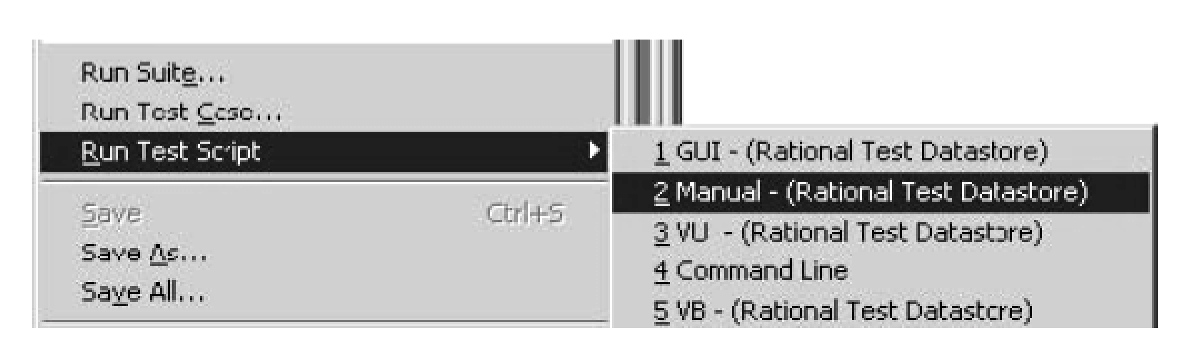


图 5-12　运行脚本

手工测试脚本运行后，会打开 Rational ManualTest 的 Run Manual Test Script 的状态，这时可以根据被测系统当前的情况，进行每一步确认，Result 中的 Checkbox 被选中，则表示此步骤已执行过，Result 显示为 Fail，表示此验证点失败，可以在 Comment 中写明失败的原因，Result 显示为 Pass，则表示此验证点通过，如图 5-13 所示。

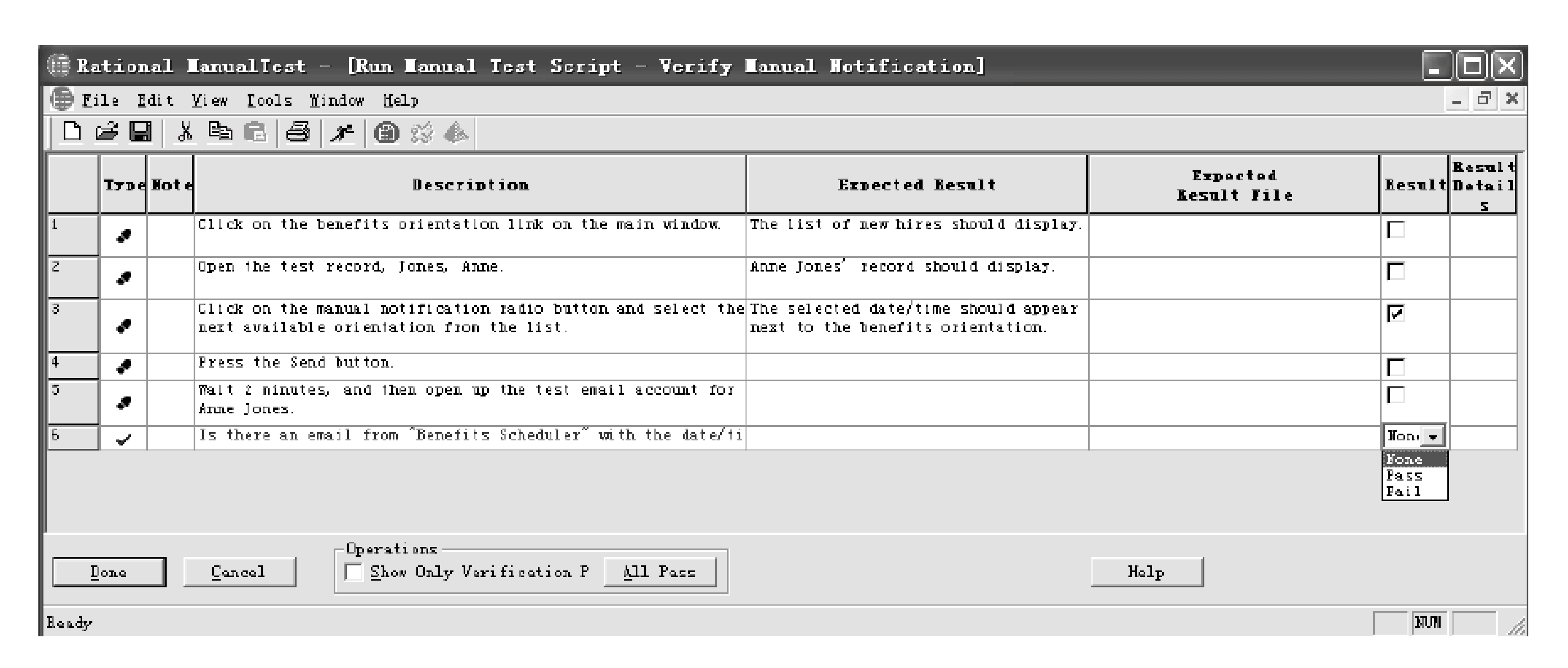


图 5-13 测试状态表

测试组(Test Suite)的功能非常强大，对于功能测试，可以将多个测试用例一起来执行，但是要保证前一个测试用例的后置条件一定是后一个测试用例的前置条件，这种测试套在测试系统流程时最有效。不过在运行 Suite 之前，要先了解有关的步骤，其中包括：检查 Suite，检查代理机，控制 Suite 运行时信息，控制 Suite 如何终止，执行 Suite 运行时的虚拟用户，配置 Suite。检查 Suite 主要是检查 Suite 中是否包含用户和计算机组，是否包含测试场景；检查代理机主要是确认代理机的实际的虚拟用户是否存在，代理机是否有效，是否可以运行，代理机的 Agent 软件是否运行；控制 Suite 运行时信息就是对 Runtime Settings 进行设置；控制 Suite 如何终止就是对 Termination Settings 进行设置，如图 5-14 所示。

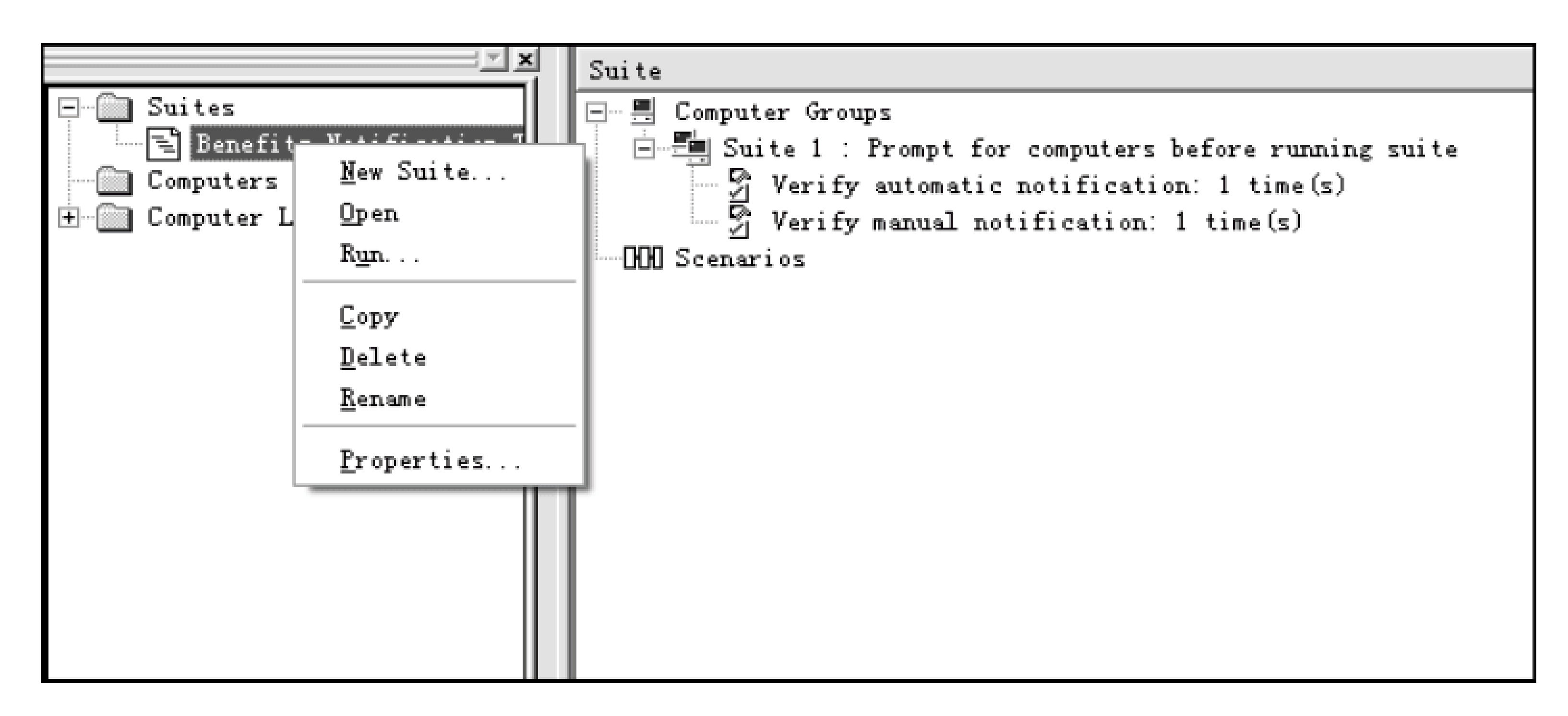


图 5-14 运行测试组

5.7.7 测试评估

通过查看测试日志，分析测试结果，并将所有的测试结果自动形成测试报告，便于分

析和统计，来确定系统是否达到标准。

当运行 Suites、测试用例和测试脚本后，TM 会自动收集测试结果，形成测试日志。每次测试执行后，都会保存最新的测试日志，在 Test Asset Workspace 的 Results 标签中，扩展 Builds 树，选择要打开的日志并打开就可以查看到测试结果。从测试日志中，可以查看到哪个测试用例是通过还是失败，并分析此测试的缺陷、系统设计的问题或其他问题。

在测试日志中，查看所有 Interpreted Result 状态为 Unevaluated 的测试，根据实际环境测试结果，可以手工更改测试状态，即人工干预测试结果，这在某种特殊情况下是非常有用的。Promoted 测试用例执行结果，即将测试用例执行结果的 Promoted 置为“选中”状态，表示此测试用例的结果是有效的、可用的，而且此测试用例结果作为统计分析使用；否则，认为是无效的，不作为有效统计。在测试日志的详细标签，单击菜单 View Properties，可以查看日志时间的详细内容。对于执行结果失败的测试用例，单击鼠标右键，在弹出的菜单中，单击 Submit Defect 菜单，即可以将此 Bug 直接提交到 ClearQuest 中。对于测试日志，也可以预览或打印出结果，如图 5-15 和图 5-16 所示。

Suite: Benefits Notification
Build: Build 1
Log Folder: Default
Iteration:
Start Date/Time: 2008-8-18 21:53:45
End Date/Time: 2008-8-18 21:54:56

Name	Actual Result	Interpreted Result	Promoted	Defects
Verify automatic notif	Pass	Pass	☐	
Verify manual notifica	Fail	Fail	☐	

图 5-15　测试日志

Event Type	Result	Date & Time	Failure Reason	Computer ...
Suite Start (Benefits Not...	Fail	2008-8-18 21:53:45	Executable Error	姚登峰
Computer Start (Suite...	Fail	2008-8-18 21:53:46		姚登峰
TestCase Start (Ve...	Pass	2008-8-18 21:53:46		姚登峰
Script Start (V...	Pass	2008-8-18 21:53:46		姚登峰
Manual Step...	Completed	2008-8-18 21:54:16		姚登峰
Manual Step...	Completed	2008-8-18 21:54:16		姚登峰
Verificatio...	Pass	2008-8-18 21:54:16		姚登峰
Script End ...	Pass	2008-8-18 21:54:16		姚登峰
TestCase End (V...	Pass	2008-8-18 21:54:16		姚登峰
TestCase Start (Ve...	Fail	2008-8-18 21:54:16		姚登峰
Script Start (V...	Fail	2008-8-18 21:54:16		姚登峰
TestCase End (V...	Fail	2008-8-18 21:54:50		姚登峰
Computer End	Fail	2008-8-18 21:54:50		姚登峰
Suite End (Benefits N...	Fail	2008-8-18 21:54:56	Executable Error	姚登峰

图 5-16　测试日志清单

TM 可以建立多种测试报告，报告的形式也很灵活，其中可以创建各种各样的查询和显示格式的报告，这些报告可以看出测试成果是否覆盖所有的软件需求，TM 也提供了集中默认种类的测试报告，其中包括测试用例报告、清单报告和性能测试报告。

测试用例报告可以跟踪测试用例的计划、实现和执行的全过程，这些报告有多种显示格式，其中包括棒图、层叠图、区域图、线图、饼图和树型图。主要的测试报告有 3 种：

第一种是测试用例分布报告,可以查看到测试计划是否覆盖了需要测试的需求和测试输入的信息,如计划的测试用例数、谁建立的、配置信息是什么、测试用例是手工实现还是脚本实现等。通过这个报告,可以确定以下信息:计划的测试用例数、用脚本实现的测试用例数、没有实现的测试用例数、用手工或自动脚本实现的测试用例数、带有迭代和配置信息的测试用例数和被怀疑的测试用例数,如图 5-17 所示。

	Planned Test Cases	Implemented Tes...	% Test Cases Implemented
Application Requirements	8	2	25
Automatic benefits orientation scheduling	4	0	0
Manual benefits orientation scheduling	1	0	0
Automatic employee notification of orientation	1	1	100
Manual employee otification of orientation	1	1	100
Record employee attendance	1	0	0

图 5-17 测试用例分布报告

第二种是测试用例结果分布报告,可以查看到带有测试结果的测试用例报告,其中包括:测试脚本、测试用例和 Suite 执行结果的信息,通过这些信息可以评估出软件质量和测试的进程,如图 5-18 所示。

	Planned Test Cases	Implemented Tes...	% Test Cases Implemented
Application Requirements	8	2	25
Automatic benefits orientation scheduling	4	0	0
Verify automtic orientation scheduling - Standard - Win2000	Yes	No	No
Verify automtic orientation scheduling - Standard - WinNT	Yes	No	No
Verify automtic orientation scheduling - Standard - Windows XP	Yes	No	No
Verify automtic orientation scheduling	Yes	No	No
Manual benefits orientation scheduling	1	0	0
Automatic employee notification of orientation	1	1	100
Manual employee otification of orientation	1	1	100
Record employee attendance	1	0	0

图 5-18 测试用例结果分布报告

第三种是测试用例趋势报告,TM 提供了几种默认的测试用例报告定义,根据这些报告定义,可以快速建立多种类的测试用例覆盖报告,通过这些报告可以跟踪测试计划、测试开发和测试执行的过程。

5.8 小　　结

本文对通用的测试管理和测试管理自动化进行了初步探讨,通过测试工具实例讲述了 TM 是如何做到测试管理自动化的。软件测试管理自动化势在必行,所以将来在实际的应用中,需要继续总结和完善"软件测试管理自动化"理论,以便能够更好、更有效、更灵活地解决软件测试管理中遇到的问题。

TM 是一种软件测试管理的自动化工具,在 RUP 的思想指导下,更能发挥它的特长。TM 可以和多种测试工具集成,通过开放的 API,可以让其他的测试工具与之交互,可以更好地发挥其强大的功能。TM 在整个测试管理的过程中,利用测试输入与测试用例关联、测试用例与测试执行关联、测试执行与测试缺陷关联,能跟踪需求的变更,及时地调整测试计划,新增或修改测试用例,重新执行测试用例,重新评估测试缺陷,这样的好处就是

在一种可控的状态下，能更有效地完成测试任务。

习题与思考

1. 什么是测试管理？
2. 测试管理包含哪几个阶段？
3. 什么是质量度量？
4. IBM 公司对软件测试管理提出了什么建议？
5. 简述软件测试流程。
6. 测试过程混乱的原因是什么？
7. 叙述“软件测试管理自动化”模型。
8. 叙述 TestManager。
9. 测试计划要解决哪几个问题？
10. 简述测试评估的工作过程。

第6章 单元测试

一则报道

2007年5月18日消息:国内信息安全商瑞星公司今日下午发出红色警报,称诺顿杀毒软件升级最新的病毒库后,会把Windows XP的关键系统文件当作病毒清除,重启后系统将会瘫痪。该次误杀只发生在简体中文版的Windows XP系统上,对国外用户几乎没有影响。

据瑞星公司称,安装了MS06-070补丁的Windows XP系统,如果将诺顿升级最新病毒库,则诺顿杀毒软件会把系统文件netapi32.dll、lsasrv.dll隔离清除,从而造成系统崩溃。由于国外品牌的笔记本和台式机多数预装了Windows XP系统和诺顿杀毒软件,这些用户极其容易遭到此次“误杀”攻击。因此中国将有数百万台电脑面临崩溃的危险。据瑞星公司称,截至中午12点已有超过7千名个人用户和近百家企业用户向瑞星客户服务中心求助。人们把它称为“赛门铁克误杀门”事件。

关于该事件的原因,瑞星研发负责人刘刚说,近年来部分反病毒企业为了追求病毒数量、查杀率、新病毒反应时间等单项技术指标,而降低了产品测试的标准,这样做会导致两个很严重的后果,一是误报率急速上升,二是容易出现产品的重大Bug,它比普通的病毒攻击所造成的后果更为严重。

“赛门铁克误杀门”事件在一片争议声中落下了帷幕,但是它身后隐蔽的问题还远未结束。诺顿误杀事件也向广大的程序员敲响了警钟,不做单元测试的程序员,在未来的发展中是没有出路的。著名测试专家Boris Beizer博士认为:“软件开发历史上最臭名昭彰的错误都是单元错误——即通过适当地单元测试就可以发现的错误。”他引证了Voyager的错误(将探测器发送到太阳)、AT&T和DCS的错误(曾造成美国三分之一的电话瘫痪),其实这些都是能够通过全面的单元测试排除掉的。这里就单元测试做了分析,并介绍了单元测试的定义,单元测试能带来的好处。

单元测试的主要目的是获取应用程序中可测试软件的最小片段,将其同代码的其余部分隔离开来,然后确定它的行为是否与预期的一样。单元测试并不能保证程序是完美无缺的,但是在所有的测试中,单元测试是第一个环节,也是最重要的一个环节。单元测试的对象是软件设计中的最小单位——模块,它是一种程序员对自己的代码进行自测试的工作,其测试依据就是软件模块的详细设计文档。单元测试通常采用白盒测试的方式,白盒测试也称结构测试或逻辑驱动测试,已知产品内部工作过程,通过测试来检测内部动作是否正常。测试按照程序内部结构进行,检验程序中的每条通路是否正确工作,而不顾

它的功能。测试是从代码的路径结构和内部逻辑信息设计测试用例，并覆盖全部代码、分支、路径、条件。所以，单元测试的一个很重要的指标就是代码覆盖率，很多软件开发标准化组织都对单元测试的代码覆盖率有明确的要求，低于标准就意味着单元测试不通过。

本章重点要掌握单元测试的策略，理解内存错误的原理。

6.1 单元测试基础

软件测试是软件构建过程中非常重要的一环，测试可以完成许多事，但最重要的是可以衡量正在开发的软件质量。RUP 认为在开发周期中，越早使测试成为投入的一部分越好。一个软件从代码编写开始，就要面临各种各样的测试，即单元测试、集成测试、回归测试等，其中与软件开发人员关系最紧密的就是单元测试。

6.1.1 什么是单元测试

单元测试是由开发者编写一小段代码，来检验被测代码的一个很小的、很明确的功能是否正确。通常说一个单元测试是用于判断某个特定条件(或者场景)下某个特定函数的行为。例如程序员可能会从字符串中删除不匹配某种模式的字符，然后确认字符串确实不再包含这些字符了。

在结构化程序时代，单元测试所说的单元是指函数。而在面向对象系统领域，对于单元测试中所谓“单元”有两种不同的解释：一是把面向对象语言中类的函数(方法)定义作为测试的单元，二是把类作为测试的单元。从实践来看，把类作为测试单位，复杂性高，可操作性较差，因此主张仍然以函数作为单元测试的测试单位，但是可以用一个测试类来组织某个类的所有测试函数。虽然是在面向对象系统中进行测试，但是不应过分强调面向对象，因为局部代码仍然是结构化的。

其实我们平时每天都在接触单元测试，例如工厂在组装一台电视机之前，会对每个元件进行测试，这就是单元测试。编写程序写了一个函数，除了极简单的外，总是要执行一下，看看功能是否正常，有时还要想办法输出些数据，如弹出信息窗口什么的，这也是单元测试。当然这些单元测试是不完整的，因为这样的单元测试代码覆盖率没有超过 70%，未覆盖的代码可能遗留大量的细小错误，这些错误还会互相影响，当 Bug 暴露出来的时候难于调试，大大地增加后期测试工作量和维护成本，也降低了开发商的竞争力。可以说，进行充分的单元测试是提高软件质量，降低开发成本的必由之路。

单元测试可以由程序员自己来完成，最终受益的也是程序员自己。对于程序员来说，如果养成完成代码后就对自己写的代码进行单元测试的习惯，不但可以写出高质量的代码，而且还能提高编程水平。

6.1.2 单元测试的必要性

编写代码时，首先必须保证它能够编译通过。如果是无法通过编译的代码可以认为是没有任何意义的。但即使代码通过编译，也只能说明代码的语法正确，却无法保证它的逻辑也一定正确。进行单元测试就是用来验证这段代码的执行过程和结果是否与期望的一致。有了单元测试，可以即时发现并修正代码的逻辑问题，确保代码能够准确的运行。

6.1.3 单元测试的优点

1. 提高了时间效率

在软件系统的开发过程中，一旦编码完成，开发人员更倾向于直接进行软件的集成工作，这样就能够尽快看到实际的软件运行情况了。而单元测试往往被看作是一项"障碍"，它推迟了对整个系统进行联调的启动时间。但忽视了单元测试的软件开发过程，却常常会发生这样的问题：大量的时间被花费在跟踪那些包含在独立单元里的简单的程序缺陷上，反而延迟了系统集成的预计时间，而且当系统投入使用时也无法确保它的可靠运行，表面上节省的时间掩盖了即将耗费的大量时间。

在实际工作中，进行完整计划的单元测试和编写实际的代码所花费的精力大致相当。一旦完成了单元测试工作，很多程序缺陷都被纠正了，在确信拥有稳定可靠的模块情况下，开发人员能够进行更高效的系统集成工作。所以说完整计划下的单元测试是对时间的更高效的利用，这才是真实意义上的节省时间。

图 6-1 摘自《实用软件度量》(Capers Jones，McGraw-Hill，1991)，它列出了在 3 个不同测试阶段，即单元测试、集成测试和系统测试所花费的时间(以一个功能点为基准)，从而可以很直观地看出单元测试的时间效率大约是集成测试的 2 倍、系统测试的 3 倍。这个图表并不表示开发人员不应该进行后阶段的测试活动，这些测试活动仍然是必须的。它们的作用是尽可能早地排除尽可能多的 Bug，以减少后阶段测试和修复错误的时间。

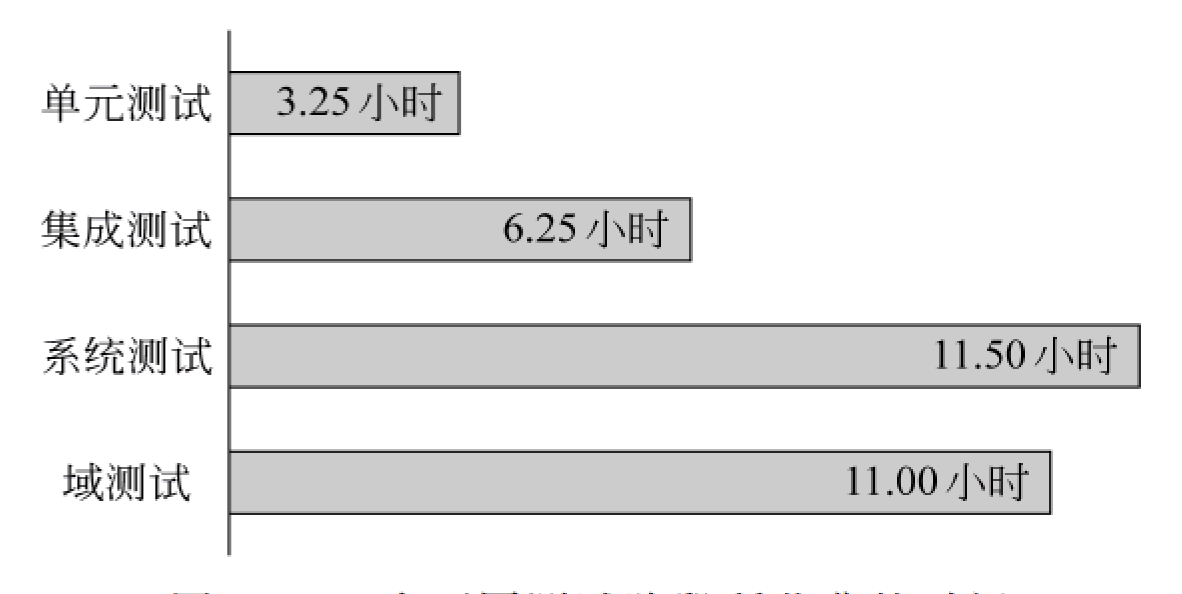


图 6-1 4 个不同测试阶段所花费的时间

注：域测试(Field Test)意思是在软件投入使用以后，针对某个领域所作的所有测试活动。

2. 验证了软件功能与详细设计说明的一致性

编译器的检查工作是基于已写好的代码的基础上，它的作用只是表明了程序可以正常运行，而无法验证程序的实际功能是否与详细设计说明的要求相符。

而单元测试中的动态执行跟踪阶段，正是以详细设计说明为基础，借助相应的单元测试工具，通过设计单元测试用例，执行待测程序，来跟踪比较实际结果与预期结果的差异。动态执行跟踪，通常分为黑盒测试与白盒测试。通过黑盒测试可以验证每个实现了的程序功能是否符合详细设计说明的要求。白盒测试可以对每个单元的内部作跟踪检查测试，通过测试验证每种内部操作是否符合详细设计说明的要求。

单元测试除了验证软件功能与详细设计说明的一致性外，还有一个作用，就是进一步保证了详细设计说明的质量。因为测试分析和测试用例设计需要依据详细设计说明来进行，这实际上也是对详细设计说明的重新检测。在这个过程中，也会发现详细设计说明评审中所没有发现的问题。

3. 确保集成测试的有效性

许多开发人员认为集成测试将会抓住所有的程序缺陷，他们却忽视了单元测试的重要性。代码集成的规模越大，意味着其复杂性越高。如果软件的被集成单元没有事先进行测试，开发人员很可能会花费大量的时间，仅仅是为了使软件能够运行，而任何实际的测试方案都无法执行。

一旦软件可以运行了，开发人员又要面对这样的问题：在考虑软件全局复杂性的前提下对每个单元进行全面的测试。这是一件非常困难的事情，甚至在创造一种单元调用测试条件时，要全面考虑单元被调用时的各种入口参数。在软件集成阶段，对单元功能全面测试的复杂程度远远超过独立进行的单元测试过程。最后的结果是测试将无法达到它所应该具有的全面性。

而单元测试的工作点放在程序的基本组成部分上，首先保证每一个单元测试通过，才能在下一步把单元集成成部件时，保证其测试的正确性。单元是整个软件的构成基础，像硬件系统中的零部件一样，只有保证零部件的质量，整个设备的质量才有保证。单元的质量也是整个软件质量的基础。

4. 提高了成本效率

在科学的软件工程规范中，程序无论什么时候做出修改，都要进行完整的回归测试。在生命周期中，尽早地对软件产品进行测试，将使其成本效率和质量得到最好的保证。

在各种测试手段中，单元测试是一种非常高效的测试方法，并且是软件测试周期中第一个进行的测试，从成本角度考虑，缺陷发现得越早，修复它所花费的成本也就越低。加强单元测试的力度，有利于降低缺陷定位和修复难度，从而降低缺陷解决成本。同时加强单元测试，也减轻了后续集成测试和系统测试的负担。比起复杂度较高的集成测试，或是维护不稳定的软件系统来，单元测试所需的费用是最低的。因为相对来说，单元测试的创建简单，维护更容易，而且可以方便地进行重复。

从全程的费用来考虑，据统计，如果一个程序缺陷，在单元测试阶段被发现并修复花费是 1 的话，到集成测试就变为 10，到系统测试就达到 100，而到实际推向市场后就会高达 1000。因此重视了单元测试也就相应地提高了软件开发的成本效率。

在提供经过测试的单元的情况下，系统的集成过程将会大大简化，开发人员可以将精力集中在单元之间的交互作用和全局的功能实现上，在花费更低开发费用的情况下，得到更稳定的软件，提高了成本效率。使软件开发的效率和软件产品的质量得到最好的保障。

6.1.4 测试的内容

单元测试的对象是软件设计的最小单位——模块或函数，单元测试的依据是详细设计说明。测试者要根据详细设计说明书和源程序清单，了解模块的 I/O 条件和模块的逻辑结构。主要采用白盒测试的测试用例，辅之以黑盒测试的测试用例，使之对任何合理和不合理的输入都能鉴别和响应。要求对所有的局部和全局的数据结构、外部接口和程序代码的关键部分进行桌面检查和代码审查。如图 6-2 所示，在单元测试中，需要对下面 5 个方面的内容进行测试，也是构造测试用例的基础。

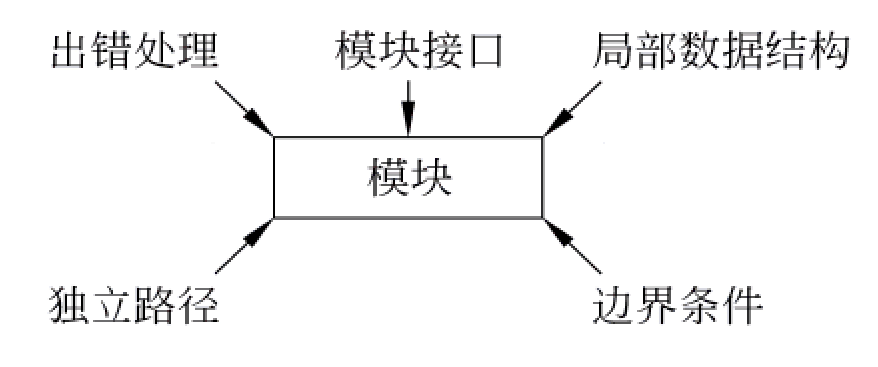


图 6-2 模块逻辑条件

1. 模块接口

接口的错误是单元测试首要考虑的方面。与结构化开发方法相比，封装避免了使用全局数据域引起的错误。但是，面向对象程序具有多个与外界发生关系的方法，会有更多的接口。因此接口的测试更加重要。

一般接口测试意味着测试模块的数据流。如果数据不能正确地输入和输出，就谈不上进行其他测试。因此，对于模块接口需要如下的测试项目：

- 调用所测模块时的输入参数与模块的形式参数在个数、属性、顺序上是否匹配；
- 所测模块调用子模块时，它输入子模块的参数与子模块的形式参数在个数、属性、顺序上是否匹配；
- 是否修改了只做输入用的形式参数；
- 输出给标准函数的参数在个数、属性、顺序上是否匹配；
- 全局变量的定义在各模块中是否一致；
- 限制是否通过形式参数来传送。

2. 局部数据结构

对于模块而言，局部数据结构是常见的错误源，这些错误在类中同样存在。因为类中的局部数据、属性对类中的每个方法而言不是局部的，因此需要针对每个方法以及方法间的协作进行测试。

模块的局部数据结构测试应设计测试用例，以检查以下各种错误：

- 检查不正确或不一致的数据类型说明；
- 使用尚未赋值或尚未初始化的变量；
- 错误的初始值或错误的默认值；
- 变量名拼写错误或书写错误；
- 不一致的数据类型。

3. 独立路径

执行路径测试由错误的控制流而造成的软件失效在类中依然存在，因此路径测试在面向对象的单元测试中仍然需要，但由于类中的方法比较简单，所以错误发生率比较低，如果方法之间存在互相调用或重载，同一段代码在不同情况下就有不同的执行通路，这种情况应该重点考虑。

对基本执行路径和循环进行测试会发现大量的错误。根据白盒测试和黑盒测试用例设计方法设计测试用例。设计测试用例查找由于错误的计算、不正确的比较或不正常的控制流而导致的错误。

常见的不正确的计算有：

- 运算的优先次序不正确或误解了运算的优先次序；
- 运算的方式错误（运算的对象彼此在类型上不相容）；
- 算法错误；
- 初始化不正确；
- 运算精度不够；
- 表达式的符号表示不正确等。

常见的比较和控制流错误有：

- 不同数据类型的比较；
- 不正确的逻辑运算符或优先次序；
- 因浮点运算精度问题而造成的两值比较不等；
- 关系表达式中不正确的变量和比较符；
- “差 1 错”，即不正确地多循环或少循环一次；
- 错误的或不可能的循环终止条件；
- 当遇到发散的迭代时不能终止循环；
- 不适当地修改了循环变量等。

4. 出错处理

比较完善的模块设计要求能预见出错的条件，并设置适当的出错处理对策，以便在程序出错时，能对出错程序重新做安排，保证其逻辑上的正确性。这种出错处理也是模块功能的一部分。例如在边界上出现错误是常见的，软件常常在它的边界上失效，类中也是如此。对这些地方要仔细地选择测试用例，认真加以测试。

表明出错处理模块有错误或缺陷的情况有：

- 出错的描述难以理解；

- 出错的描述不足以对错误定位和确定出错的原因；
- 显示的错误与实际的错误不符；
- 对错误条件的处理不正确；
- 在对错误进行处理之前，错误条件已经引起系统的干预；
- 如果出错情况不予考虑，那么检查恢复正常后模块可否正常工作。

5. 边界条件

边界上出现错误是常见的。设计测试用例检查：

- 在 n 次循环的第 1 次、第 2 次、第 n 次是否有错误；
- 运算或判断中取最大值最小值时是否有错误；
- 数据流、控制流中刚好等于、大于、小于确定的比较值时是否出现错误。

6.1.5 测试的环境构成

由于单元函数不能独立运行，需要构造一个运行环境才能完成测试，单元测试环境由驱动函数、被测试单元函数和桩函数组成，如图 6-3 所示。

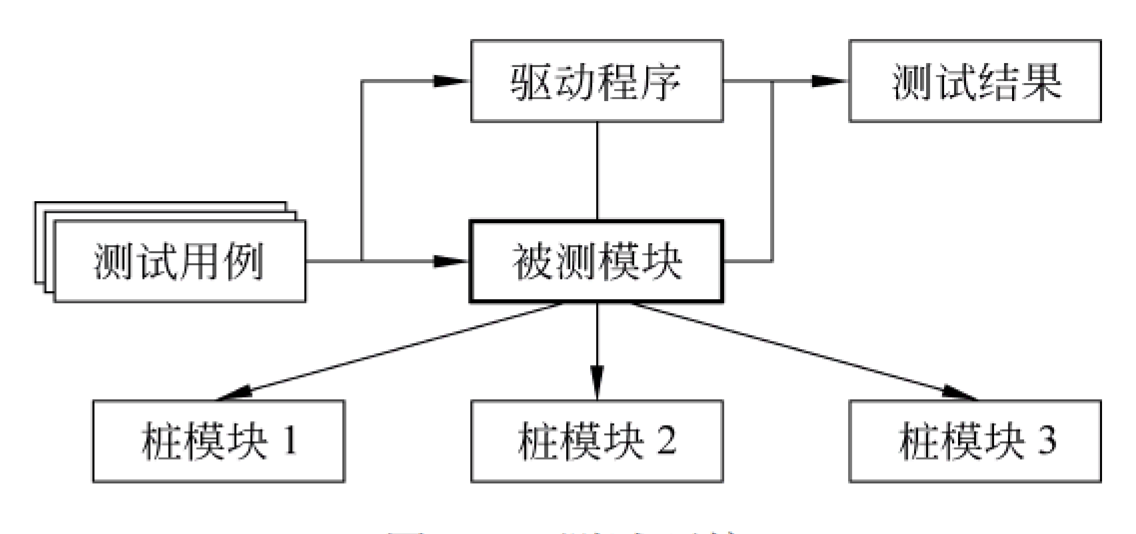


图 6-3 测试环境

单元测试在编码阶段进行。在源程序代码编制完成、经过评审和验证、确认没有语法错误之后，就可以开始进行单元测试的测试用例设计。要利用软件设计文档，设计可以验证程序功能、找出程序错误的多个测试用例。

对于每一组输入，应该有预期的正确结果。在单元测试时，如果模块不是独立的程序，需要辅助测试模块，这里介绍两种辅助模块：

① 驱动模块(driver)：所测模块的主程序。它接收测试数据，把这些数据传递给所测试模块，最后再输出测试结果。当被测试模块能完成一定功能时，也可以不要驱动模块。

② 桩模块(stub)：用来代替所测模块调用的子模块。

驱动函数是一个主函数，其作用是将测试用例数据传送给被测试单元函数，并引导被测试单元函数运行，保存运行结果。桩函数是一个构造子函数，用来替代被测试单元函数调用的子函数，是为了隔离子函数对被测试单元函数的影响而构造的，它可以是一个“空”子函数或是对接口做少量操作的子函数。有时在一个单元函数中可以构造多个桩函数。如果被测试单元函数没有调用的子函数或被调用的子函数，对被测试单元函数结果没有

影响，可以不用构造桩函数。

6.2 单元测试策略

白盒测试和黑盒测试是软件测试方法的两大分类。这里所要介绍的单元测试策略既有白盒测试范畴内的，也有黑盒测试范畴内的。

6.2.1 使用白盒测试技术的单元测试

白盒测试也称结构测试或逻辑驱动测试，它是按照程序内部的结构进行测试。检测产品内部动作是否按照设计规格说明书的规定正常进行，检验程序中的每条通路是否都能按预定要求正常工作。在白盒测试中，测试人员把测试对象看作一个打开的盒子，测试人员依据程序内部逻辑结构相关信息，设计或选择测试用例，对程序所有逻辑路径进行测试，通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致。

白盒测试可采用静态测试和动态测试，静态测试不执行程序，通过人工或工具对代码进行检查，动态测试可通过输入一组预先按照一定的测试准则构造的实例数据，来动态运行程序，代码走查是一种常用的白盒测试方法。

关于代码走查，目前也有许多种做法。有的代码走查采用工具走查和人工走查两种方式。实践证明，这两种方式均简单、高效。

1. 工具走查

工具走查是借助测试工具对代码进行检查，主要用于发现一些编码规范方面的问题。比较经典的测试工具都带有几百条预先定义的代码规范。测试者可根据需要有选择地使用工具对这些代码规范进行检查。此外还可根据自己部门的实际需要，定制一些新的编码规范。编码规范的检查可提高程序的可读性、可维护性、可测性和稳定性，也可早期发现一些程序中存在的重大隐患和错误。这些隐患或缺陷如果前期没有排除，在后期集成为一个软件之后，在进行功能测试的时候，往往会引起一些严重的软件异常，如软件崩溃等。此时查找和排除缺陷工作变得耗时耗力，效率低下，问题难以定位。而通过工具走查这些问题可直接定位到代码行，在前期进行排除，且一个几千行的程序模块走查一次只需要几分钟，测试人员只需要花费少量时间，对测试结果进行分析提取即可，效率极高。

一般的测试工具都支持至少几百条编码规范。这些编码规范，在不同的情况下是不完全适应的，测试人员应根据自己部门的要求，合理选择编码规范。因为许多编码规范违反以后并不会立即引起程序错误，而管理人员在项目开发早期也未制定相应的强制性要求，考虑工期和人员开支等情况，测试人员可能只将一部分违反编码规范的走查结果列为必须修改的错误。如 C++ Test 中的 Must Have Rules，这里面的规则都是诸如新建之后未删，变量未初始化，这样就可能存在严重的隐患问题。C++ Test 的规则中还包括 Security Rules/Nice to Have Rules 等，测试者可根据自己部门的实际情况选用，也可自

行添加一些工具中没有的规则。一般的白盒测试工具都支持编码规范的自定义。

2. 人工走查

人工走查也是一种简单有效的走查方法。走查之前成立走查小组，负责人提前派发设计说明、编码任务书和问题检查单等材料。小组成员阅读材料，了解被走查单元的详情后召开走查会议，程序员讲解程序，在这个过程中走查组成员对代码进行检查，提出问题，进行讨论，发现程序错误。这是常用的人工代码走查方法，是一种静态的测试方法。

另外一种比较有效的是经过改进之后的人工走查方法，它是一种动态的测试方法。在走查会议上，程序不是由程序员进行静态讲解，而是一边讲解一边进行动态的运行。在这之前，测试人员为所测程序准备一组有代表性的用例，走查的时候，程序员在讲解的同时，按测试人员的用例运行程序，走查人员可以对程序的每一步运行状态进行记录，对程序运行逻辑进行检查和验证。这种方式通常可以发现工具走查所不能发现的代码逻辑问题。在针对某个局部问题进行讨论的时候，有可能还会牵出与之关联的其他问题，甚至推翻现有的设计，导致需求和设计的重定义，极大地改进了软件的质量。在走查中通常选择有着多年软件设计和编码经验的“高手”组成走查组。因此在走查过程中，常常还会发现一些通过其他手段很难发现的程序隐患。这些隐患往往在特定的条件下才会被触发，但一旦发生，后果却很严重。即使在软件测试阶段发现了，但开发人员也无法重现，难以定位。这种人工走查方式的优点是可以发现绝大部分代码逻辑和隐蔽问题，但走查效果很大程度上依赖于走查组人员的经验积累。

纯粹的白盒测试关注的是程序的内部实现，不关注程序的外在表现，经过白盒测试的代码不能证明该单元真正完成了需求中规定的功能。白盒测试的关注点是代码本身，而非代码所实现的功能，所以经过走查的代码可以明显地提高代码质量，但还不能确定代码所实现的功能是否完全符合要求。这时就需要对代码进行功能测试。

6.2.2 使用黑盒测试技术的单元测试

黑盒测试也叫功能测试，是通过测试来检测每个功能是否都能正常使用。黑盒测试方法的着眼点是单元模块的功能。通过设计功能测试用例，编写驱动程序，运行单元的各功能，可发现软件界面和软件功能方面的问题。此时设计用例可参考需求说明和设计说明中对该功能模块的功能和外部接口的定义，把程序作为一个黑盒子，不考虑程序的内部结构和内部特性，在程序接口处进行测试，检查程序是否按照需求规格说明的规定正常使用，在接收输入后是否能产生符合要求的输出。

黑盒测试用例设计是根据程序的功能进行的。程序的功能定义存在一些人为的和主观的因素，常常是不全面的；各个输入数据和操作次序之间也有很多种组合关系，有些组合可能会产生问题，无法保证这些组合都经过了测试。

很显然，普通的黑盒测试存在着局限性，很难衡量黑盒测试的完整性。因此引入了白盒测试中的逻辑覆盖指标来标识测试结果。逻辑覆盖是白盒测试中的测试覆盖率计算方法，包括语句覆盖、判定覆盖、条件覆盖、条件判定组合覆盖、多条件覆盖、修正判定条件覆

盖以及路径覆盖。这几种覆盖对测试覆盖的强度依次增强。考虑人工和投入产出比等情况，通常对单元测试的要求是语句覆盖达到100%，判定覆盖达到100%。E. F. Miller发现，当一组测试用例满足判定覆盖时，可以发现全部缺陷中的大约85%。

这样形成的功能测试方法，是一种介于黑盒和白盒之间的灰盒测试方法。其过程是在对单元的功能进行测试的时候，先从功能的角度设计测试用例，对单元的功能进行测试。测试的同时借助工具监测代码覆盖率，在用例执行完成之后查找没有被覆盖的语句和分支，有针对性地设计用例，覆盖它们。这样能达到要求的测试完整性，又能避免不必要的重复。

这种灰盒测试方式比单纯的黑盒测试更有效、更完整。测试工具一般支持覆盖率的实时显示，并能标识还未被执行的语句和分支，在执行功能测试的时候，可以通过测试工具得到想要的逻辑覆盖情况。

6.2.3 策略的选择

综合以上单元测试方法的介绍，可以看到工具代码走查、人工代码走查和功能测试这3种方式都是很有效的，各有其优缺点。

通过白盒测试工具对代码进行检查是一种代价很低的测试方法，选定要检查的规范后，整个检查过程只需要几分钟。针对C和C++代码，测试工具可以很好地满足测试要求，但测试工具比较昂贵，需要一定的经费支持。

人工对代码进行走查周期也很短，走查一个单元平均需要不到1小时。但人工走查主要依赖于走查组成员的个人经验和技术，走查组成员技术经验将直接影响到测试效果。所以走查之前，需要对走查会议进行良好的策划和组织。

功能测试是代价最高的测试方式。功能测试可能面临多次回归测试，周期较长，而且需要测试人员编写维护测试驱动程序和桩程序，对测试人员的编程能力也有一定的要求，除此之外对测试用例的设计和维护，也是一项很烦琐的工作。

这是目前被广泛采用的3种单元测试方法，项目经理和测试部门应根据实际情况灵活选用。最好是测试需求驱动测试方法，对重用性高、调用频繁或核心功能的单元模块，在进度允许的情况下，上述3种测试方式应该都选用，各种测试方式互为补充，最大限度地发现问题，保证单元质量。进度不允许的情况下，对非关键模块或重要性级别较低的单元，可适当选择这3种方法中的1种或2种组合进行测试。当然，这3种方法也有一定的交叉和重复，这就需要测试者明确每个测试过程的目标和范围，尽量避免重复性劳动，提高效率，减少测试资源的浪费。

任何测试都不能做到完全彻底。如果测试资源不足，测试人员为保证多个项目的测试工作，必须寻找快速而有效的测试方法。以上3种方式经过实践证明，在测试资源极为有限的情况下，也可以达到较好的测试效果。

6.2.4 日构建

日构建(Nightly Build或Daily Build)是将一个软件项目的所有最新代码取出，从头

开始编译、构建、运行测试软件包，对代码进行单元测试并对主要功能进行测试，发现错误并报告信息状态的完整过程。每日构建意味着自动地、每天完整地构建整个代码树，将构建测试、单元测试、安装测试、Bug 报告、E-mail 触发通知等结合起来。每日构建是持续集成的一个最佳实践。有些情况下在白天进行构建会消耗过多的计算机资源，对开发造成影响，因此许多每日构建是在夜间进行的，所以 Daily Build 就成了 Nightly Build。

每日构建与单元级别的开发中。越来越多的组织开始实行每日构建＋冒烟测试的构建过程。麦肯锡公司的调查发现，成功公司中有 94％在每天或至少每周完成构建，而不成功公司绝大多数每月甚至很少去做构建。用每日构建方法开发的软件有 Mozilla 等。在每日构建中，新开发的版本从源代码库中检查、编译、构建然后进行冒烟测试。只有通过自动化冒烟测试才能认为代码构建合格。发现的缺陷通过电子邮件反馈给开发者，以进行快速的修复。现在有许多工具支持每日构建，包括 Cruise Control、Visual Build。

每日构建的技术实现不是最主要的，而开发团队以每日构建来分配项目发布压力、提高代码质量的目的才是本质。每日构建的技术也可以说是协作开发中的质量保证技术和进度控制技术。围绕每日构建制定相应的团队规则，有利于每日构建发挥积极的作用，也有助于优秀团队的建设。如何更有效地实施 BVT 测试（每日构建的最后一步），将单元测试、安装测试、Bug 报告、E-mail 触发通知等结合起来，是后续要研究、实践的问题。也需要开发团队行动起来，更好地建立每日构建流程，配合良好的团队纪律，围绕每日构建所规定的规范，努力实现团队开发过程的最优化，以保证质量和进度，提高生产率。

6.3 单元测试工具实践

目前有很多单元测试工具，常见的有 Parasoft 公司的 JTest 工具、开源的 Java 单元测试工具 JUnit、IBM 公司的 PurifyPlus 等，它们都支持程序覆盖率的自动统计、问题定位等。这里以 IBM 公司的 PurifyPlus 工具为例做一下介绍。

IBM Rational PurifyPlus™能够帮助开发人员和测试人员达到项目质量的共同目标。IBM 公司的单元测试工具 Rational PurifyPlus 产品有 3 个套件：

① 内存检查工具 Purify：包括收集方法和对象层次的内存分析数据，并指出应用程序的内存热点所在。

② 代码效率分析工具 Quantify：包括收集方法和线层次的分析数据并指出应用程序的性能瓶颈。

③ 代码覆盖分析工具 PureCoverage：通过突出未执行的方法，代码行来收集没有测试到的部分的数据。这三种工具不仅对 Java，对 Visual C/C++、Visual Basic 和.NET 应用也都提供了全面的支持。

下面将依次讨论每个成分的工作方式。

6.3.1 Purify 组件

Purify 组件用来进行内存分析。应用程序运行时内存错误和泄漏是最难检测的问题之一。它可能会随机触发，并且相当难预测。Purify 能够帮助检查出这些类型的错误。Purify 非常容易使用，它有大量的命令行接口，帮助获得正确的信息。这里首先介绍基础知识——内存问题。

在 C/C++ 程序中，有关内存使用的问题是最难发现和解决的。这些问题可能导致程序莫名其妙地停止、崩溃，或者不断消耗内存直至资源耗尽。由于 C/C++ 语言本身的特质和历史原因，程序员使用内存需要注意的事项较多，而且语言本身也不提供类似 Java 的垃圾清理机制。编程人员使用一定的工具来查找和调试内存相关问题是十分必要的。

与内存有关的问题可以分成两类：内存访问错误和内存使用错误。内存访问错误指错误地读取内存和错误地写内存。错误地读取内存可能让模块返回意想不到的结果，从而导致后续的模块运行异常。错误地写内存可能导致系统崩溃。内存使用方面的错误主要是指申请的内存没有正确释放，从而使程序运行逐渐减慢，直至停止。这方面的错误由于表现比较慢，很难被人工察觉。程序也许运行了很久才会耗尽资源，发生问题。

一个典型的 C++ 内存布局如图 6-4 所示。

自底层向上，内存中依次存放着只读的程序代码和数据、全局变量和静态变量、堆中的动态申请变量和堆栈中的自动变量。自动变量就是在函数内声明的局部变量。当函数被调用时，自动变量被压入栈；当函数返回时，自动变量就要被弹出堆栈。堆栈的使用基本上由系统控制，用户一般不会直接对其进行控制，所以堆栈的使用还是相对安全的。动态内存是一柄双刃剑：它可以提供程序员更灵活的内存使用方法，而且有些算法没有动态内存很难实现；但是动态内存往往又是内存问题衍生的沃土。

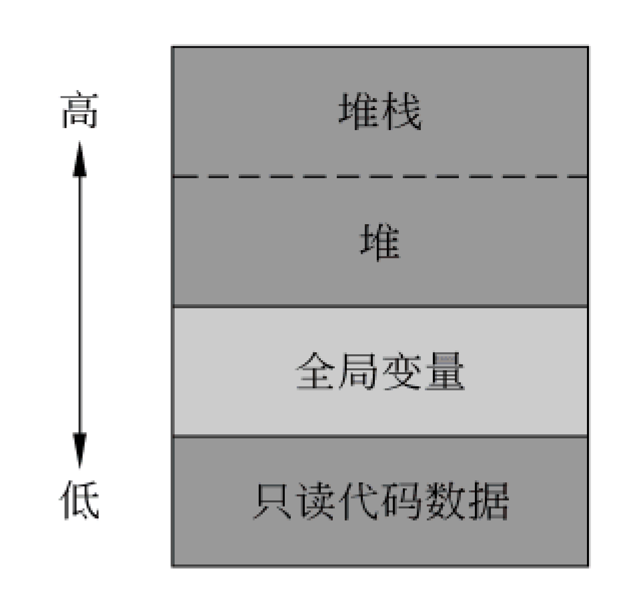


图 6-4 典型 C++ 内存区域

相对用户使用的语言，动态内存的申请一般由 malloc/new 来完成，释放由 free/delete 完成。基本的原则可以总结为：一对一，不混用。也就是说一个 malloc 必须对应一个且唯一的 free；new 对应一个且唯一的 delete；malloc 不能和 delete 对应，new 不能和 free 对应。另外在 C++ 中要注意 delete 和 delete[]的区别。delete 用来释放单元变量，delete[]用来释放数组等集聚变量。

1. 内存访问错误

可以将内存访问错误大致分成以下几类：数组越界读或写、访问未初始化内存、访问已经释放的内存和重复释放内存或释放非法内存。

下面的代码是集中显示上述问题的典型例子：

```
#include <iostream>
using namespace std;
```

```
3  int main(){
4      char * str1="four";
5      char * str2=new char[4];              //not enough space
6      char * str3=str2;
7      cout<<str2<<endl;                     //UMR
8      strcpy(str2,str1);                    //ABW
9      cout<<str2<<endl;                     //ABR
10     delete str2;
11     str2[0]+=2;                           //FMR and FMW
12     delete str3;                          //FFM
13  }
```

由以上的程序可以看到：在第 5 行分配内存时，忽略了字符串终止符"\0"所占空间导致了第 8 行的数组越界写(array bounds write)和第 9 行的数组越界读(array bounds read)；在第 7 行，打印尚未赋值的 str2 将产生访问未初始化内存错误(uninitialized memory read)；在第 11 行使用已经释放的变量，将导致释放内存读和写错误(freed memory read and freed memory write)；最后由于 str3 和 str2 所指的是同一片内存，第 12 行又一次释放了已经被释放的空间 (free freed memory)。

这个包含许多错误的程序可以编译连接，而且可以在很多平台上运行。但是这些错误就像定时炸弹，会在特殊配置下触发，造成不可预见的错误。这就是内存错误难以发现的一个主要原因。

2. 内存使用错误

内存使用错误也叫内存泄漏，它比内存访问错误更加难以发现。主要有两个原因：第一，内存使用错误是“慢性病”，它的症状可能不会在少数、短时间的运行中体现；第二，内存使用错误是因为“不作为”，即忘记释放内存，而不是“做错”造成的。这样由于忽略造成的错误在检查局部代码时很难发现，尤其是当系统相当复杂的时候。

以下这段小程序演示了堆内存发生泄漏的情形：

```
void MyFunction(int nSize)
{
char * p=new char[nSize];
if(!GetStringFrom( p, nSize ) ){
MessageBox("Error");
return;
}
…//using the string pointed by p;
delete p;
}
```

当函数 GetStringFrom()返回 0 的时候，指针 p 指向的内存就不会被释放。这是一种常见的发生内存泄漏的情形。程序在入口处分配内存，在出口处释放内存，但是 C 函数可以在任何地方退出，所以一旦有某个出口处没有释放应该释放的内存，就会发生内存

泄漏。

广义地说，内存泄漏不仅包含堆内存的泄漏，还包含系统资源的泄漏（resource leak），例如核心态 HANDLE、GDI Object、SOCKET、Interface 等，从根本上说这些由操作系统分配的对象也消耗内存，如果这些对象发生泄漏，最终也会导致内存的泄漏。而且，某些对象消耗的是核心态内存，这些对象严重泄漏时会导致整个操作系统不稳定。所以相比之下，系统资源的泄漏比堆内存的泄漏更为严重。

如下的 GDI Object 的泄漏是一种常见的资源泄漏：

```
void CMyView::OnPaint( CDC * pDC )
{
CBitmap bmp;
CBitmap * pOldBmp;
bmp.LoadBitmap(IDB_MYBMP);
pOldBmp=pDC->SelectObject( &bmp );
…
if( Something() ){
return;
}
pDC->SelectObject( pOldBmp );
return;
}
```

当函数 Something()返回非 0 的时候，程序在退出前没有把 pOldBmp 选回 pDC 中，这会导致 pOldBmp 指向的 HBITMAP 对象发生泄漏。这个程序如果长时间地运行，可能会导致整个系统花屏。这种问题在 Windows 9x 下比较容易暴露出来，因为 Windows 9x 的 GDI 堆比 Windows 2000 或 Windows NT 的要小很多。

以发生的方式来分类，内存泄漏可以分为 4 类：

① 常发性内存泄漏。发生内存泄漏的代码会被多次执行到，每次被执行的时候都会导致一块内存泄漏。例如上述的 GDI Object 的泄漏例子中，如果 Something()函数一直返回 True，那么 pOldBmp 指向的 HBITMAP 对象总是发生泄漏。

② 偶发性内存泄漏。发生内存泄漏的代码只有在某些特定环境或操作过程下才会发生。例如上述的 GDI Object 的泄漏例子中，如果 Something()函数只有在特定环境下才返回 True，那么 pOldBmp 指向的 HBITMAP 对象并不总是发生泄漏。常发性和偶发性是相对的。对于特定的环境，偶发性的也许就变成了常发性的。所以测试环境和测试方法对检测内存泄漏至关重要。

③ 一次性内存泄漏。发生内存泄漏的代码只会被执行一次，或者由于算法上的缺陷，导致总会有一块仅且一块内存发生泄漏。例如，在类的构造函数中分配内存，在析构函数中却没有释放该内存，但是因为这个类是一个 Singleton，所以内存泄漏只会发生一次。例如：

```
char * g_lpszFileName=NULL;
void SetFileName( const char * lpcszFileName )
```

```
{
if( g_lpszFileName ){
free( g_lpszFileName );
}
g_lpszFileName=strdup( lpcszFileName );
}
```

如果程序在结束的时候没有释放 g_lpszFileName 指向的字符串，那么，即使多次调用 SetFileName()，总会有一块内存，而且仅有一块内存发生泄漏。

④ 隐式内存泄漏。程序在运行过程中不停地分配内存，但是直到结束的时候才释放内存。严格地说这里并没有发生内存泄漏，因为最终程序释放了所有申请的内存。但是对于一个服务器程序，需要运行几天，几周甚至几个月，不及时释放内存，也可能导致最终耗尽系统的所有内存。所以，称这类内存泄漏为隐式内存泄漏。例如：

```
class Connection
{
public:
Connection( SOCKET s);
~Connection();
…
private:
SOCKET _socket;
…
};
class ConnectionManager
{
public:
ConnectionManager(){
}
~ConnectionManager(){
list::iterator it;
for( it=_connlist.begin(); it!=_connlist.end(); ++it ){
delete (*it);
}
_connlist.clear();
}
void OnClientConnected( SOCKET s ){
Connection* p=new Connection(s);
_connlist.push_back(p);
}
void OnClientDisconnected( Connection* pconn ){
_connlist.remove( pconn );
delete pconn;
}
private:
```

```
list _connlist;
};
```

假设在 Client 从 Server 端断开后，Server 并没有呼叫 OnClientDisconnected()函数，那么代表那次连接的 Connection 对象就不会被及时地删除(在 Server 程序退出的时候，所有 Connection 对象会在 ConnectionManager 的析构函数里被删除)。当不断地有连接建立、断开时，隐式内存泄漏就发生了。

从用户使用程序的角度来看，内存泄漏本身不会产生什么危害，作为一般用户，根本感觉不到内存泄漏的存在。真正有危害的是内存泄漏的堆积，这会最终消耗尽系统所有的内存。从这个角度来说，一次性内存泄漏并没有什么危害，因为它不会堆积，而隐式内存泄漏危害性则非常大，因为较之于常发性和偶发性内存泄漏它更难被检测到。

Purify 组件可在 C/C++ 程序中去除分解内存使用。当它发现事件比被分配时使用更多的内存，或在它初始化之前或释放之后使用内存，就会发布一个错误报告。Purify 还可以报告内存泄漏，例如，当一个程序在丢失内存跟踪之前不能释放一个或更多的内存板块时。尤其是在 Java™和 Microsoft .NET 程序中，Purify 能够帮助识别使用内存并且没有将它释放的程序区域，这是另一种类型的内存泄漏。

对于 C/C++ 程序（而不是 Java 或 .NET），Purify 使用了仪表化技术，它是指通过在这个程序执行中的每一个内存访问周围插入额外的指令来工作。当运行这个仪表化程序时，它将正常运作，进行与平时一样的操作。但是，Purify 在里面可以检查问题。如果这个程序从还没有被初始化的内存中读取，或者这个程序读取或改写已经被释放或仍然存在于分配区域之外的内存，就可以看到一个 Purify 错误报告。

有些类型的内存访问错误（例如使用一个 NULL 指示器）将会导致一个程序完全崩溃。这是一个很容易就可以辨认出的错误，可以用一个正规调试器来找到它。Purify 可以看到更多潜伏的内存错误类型，它们可能出现在程序看起来似乎运转正常的情况下。

例如，一个逻辑性的错误可能不会导致程序崩溃或测试失败，但是仍然存在内存崩溃的隐患。某天当这种内存错误由于一个不正确的操作超出了内存的范围，那么就会导致系统崩溃，最后造成客户的不满意。其实在项目早期，Purify 能够很容易发现这种类型的错误。

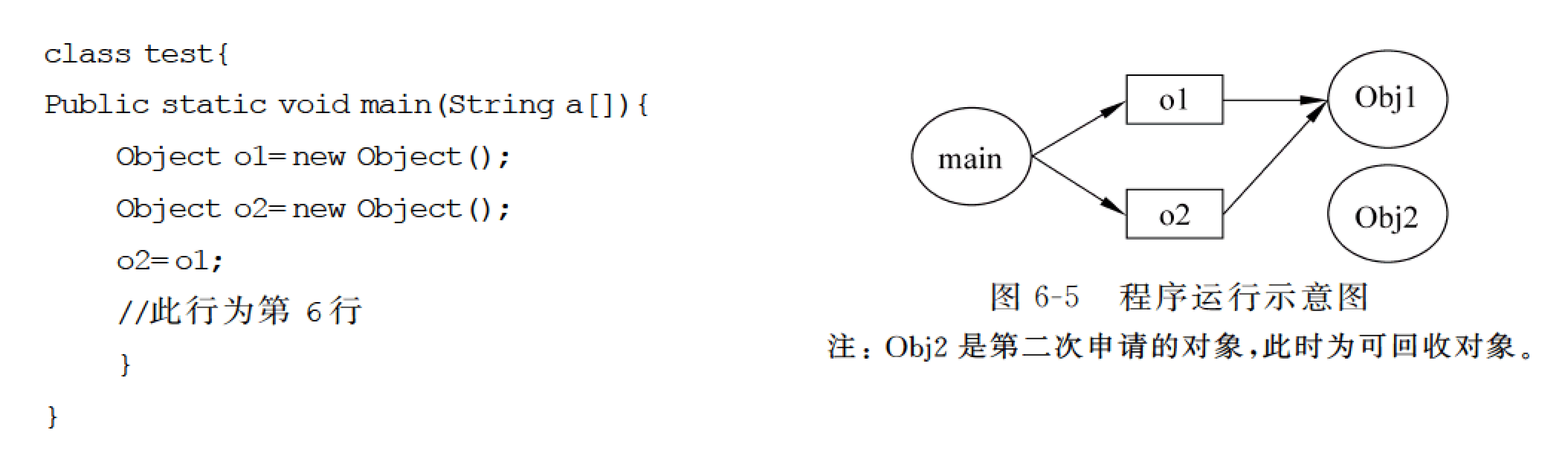
3. Java 内存错误

Java 也存在内存泄露，但它的表现与 C++ 不同。Java 的内存管理就是对象的分配和释放问题。在 Java 中，程序员需要通过关键字 new 为每个对象申请内存空间（基本类型除外），所有的对象都在堆（heap）中分配空间。另外，对象的释放是由 GC 决定和执行的。在 Java 中，内存的分配是由程序完成的，而内存的释放是由 GC 完成的，这种收支两条线的方法确实简化了程序员的工作。但同时，它也加重了 JVM 的工作。这也是 Java 程序运行速度较慢的原因之一。因为，GC 为了能够正确释放对象，GC 必须监控每一个对象的运行状态，包括对象的申请、引用、被引用、赋值等，GC 都需要进行监控。监视对

象状态是为了更加准确地、及时地释放对象，而释放对象的根本原则就是该对象不再被引用。

为了更好理解 GC 的工作原理，可以将对象考虑为有向图的顶点，将引用关系考虑为图的有向边，有向边从引用者指向被引对象。另外，每个线程对象可以作为一个图的起始顶点，例如大多程序从 main 进程开始执行，那么该图就是以 main 进程顶点开始的一棵根树。在这个有向图中，根顶点可达的对象都是有效对象，GC 将不回收这些对象。如果某个对象（连通子图）与这个根顶点不可达（注意该图为有向图），那么认为这个（这些）对象不再被引用，可以被 GC 回收。

下面举一个例子说明如何用有向图表示内存管理。对于程序的每个时刻，都有一个有向图表示 JVM 的内存分配情况。图 6-5 就是程序运行到第 6 行的示意图。



```
class test{
Public static void main(String a[]){
    Object o1=new Object();
    Object o2=new Object();
    o2=o1;
    //此行为第 6 行
    }
}
```


图 6-5　程序运行示意图

注：Obj2 是第二次申请的对象，此时为可回收对象。

Java 使用有向图的方式进行内存管理，可以消除引用循环的问题，例如有三个对象，相互引用，只要它们和根进程是不可达的，那么 GC 也是可以回收它们的。这种方式的优点是管理内存的精度很高，但是效率较低。另外一种常用的内存管理技术是使用计数器，例如 COM 模型采用计数器方式管理构件，它与有向图相比，精度行低（很难处理循环引用的问题），但执行效率很高。

在 Java 中，内存泄漏就是存在一些被分配的对象。这些对象有两个特点：首先，这些对象是可达的，即在有向图中，存在通路可以与其相连；其次，这些对象是无用的，即程序以后不会再使用这些对象。如果对象满足这两个条件，这些对象就可以判定为 Java 中的内存泄漏，这些对象不会被 GC 所回收，然而它却占用内存。

在 C++ 中，内存泄漏的范围更大一些。有些对象被分配了内存空间，然后却不可达，由于 C++ 中没有 GC，这些内存将永远收不回来。在 Java 中，这些不可达的对象都由 GC 负责回收，因此程序员不需要考虑这部分的内存泄露。

通过分析得知，对于 C++，程序员需要自己管理边和顶点，而对于 Java 程序员只需要管理边就可以了（不需要管理顶点的释放）。通过这种方式，Java 提高了编程的效率。

在 Java 中也有内存泄漏，但范围比 C++ 要小一些。因为 Java 从语言上保证任何对象都是可达的，所有的不可达对象都由 GC 管理。

对于程序员来说，GC 基本是不可见的。虽然只有几个函数可以访问 GC，例如运行 GC 的函数 System.gc()，但是根据 Java 语言规范定义，该函数不保证 JVM 的垃圾收集器一定会执行。因为，不同的 JVM 实现者可能使用不同的算法管理 GC。通常，GC 的线程的优先级别较低。JVM 调用 GC 的策略也有很多种。有的是内存使用到达一定程度

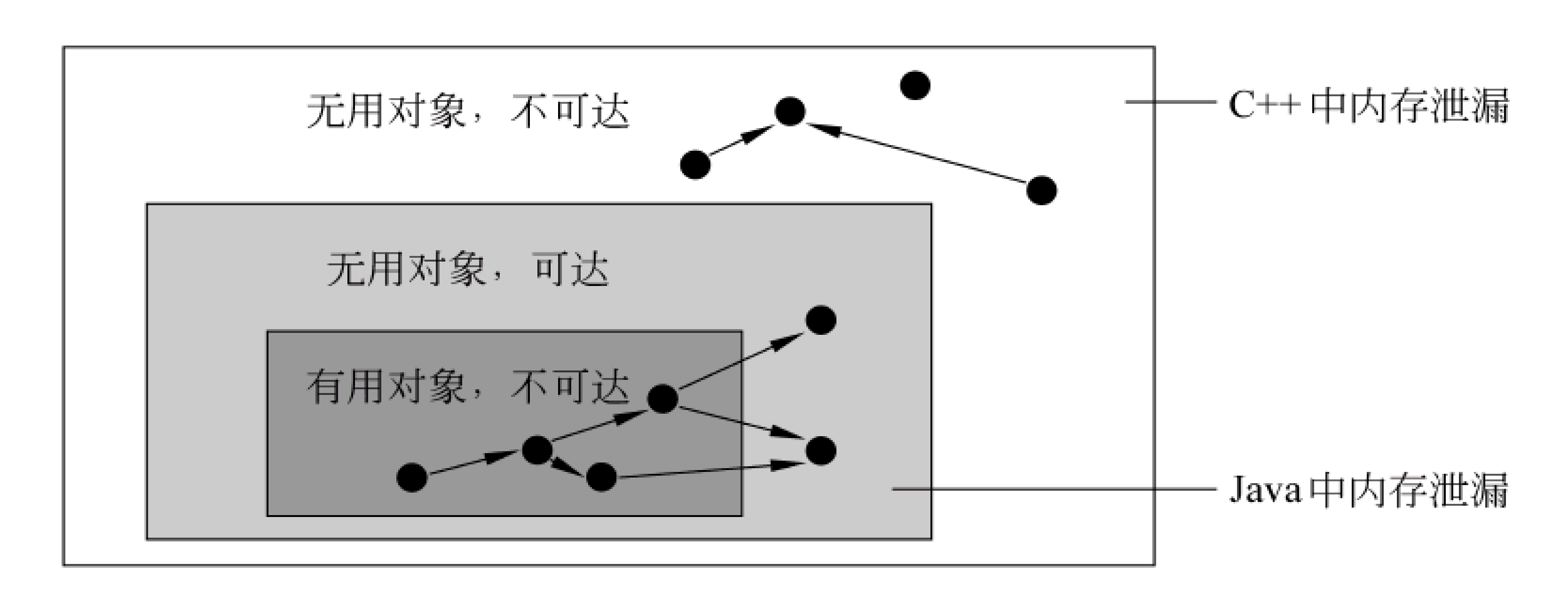


图 6-6　Java 内存泄露

时，GC 才开始工作，有的是定时执行的，也有的是平缓执行 GC，还有的是中断式执行 GC。但通常来说用户不需要关心这些。除非在一些特定的场合，GC 的执行影响应用程序的性能，例如对于基于 Web 的实时系统，如网络游戏等，用户不希望 GC 突然中断应用程序执行而进行垃圾回收，那么需要调整 GC 的参数，让 GC 能够通过平缓的方式释放内存，例如将垃圾回收分解为一系列的小步骤执行，Sun 提供的 HotSpot JVM 就支持这一特性。

下面是一个简单的内存泄露的例子：

```
Vector v=new Vector(10);
for (int i=1;i<100; i++)
{
Object o=new Object();
v.add(o);
o=null;
}
//此时，所有的 Object 对象都没有被释放，因为变量 v 引用这些对象
```

在这个例子中，循环申请 Object 对象，并将所申请的对象放入一个 Vector 中，如果仅释放引用本身，那么 Vector 仍然引用该对象，所以这个对象对 GC 来说是不可回收的。因此，如果对象加入到 Vector 后，还必须从 Vector 中删除，最简单的方法就是将 Vector 对象设置为 null。

这种类型的内存泄漏将会导致程序越来越大，运行越来越慢，最终耗尽内存。对于 Java 和.NET 程序，当项目仍然在使用本应该已经释放的内存时，Purify 能够帮助识别出来。Purify 工作原理是通过监测 Java 程序运行时，所有对象的申请、释放等动作，将内存管理的所有信息进行统计、分析、可视化。开发人员将根据这些信息判断程序是否有内存泄漏问题。

4. Purify 原理

程序运行时的分析可以采用多种方法。Purify 使用了具有专利的目标代码插入技术(object code insertion，OCI)。它在程序的目标代码中插入了特殊的指令用来检查内存的状态和使用情况。这样做的好处是不需要修改源代码，只需要重新编译就可以对程序进

行分析。

对于所有程序中使用的动态内存，Purify将它们按照状态进行归类。这可以由图6-7来说明。

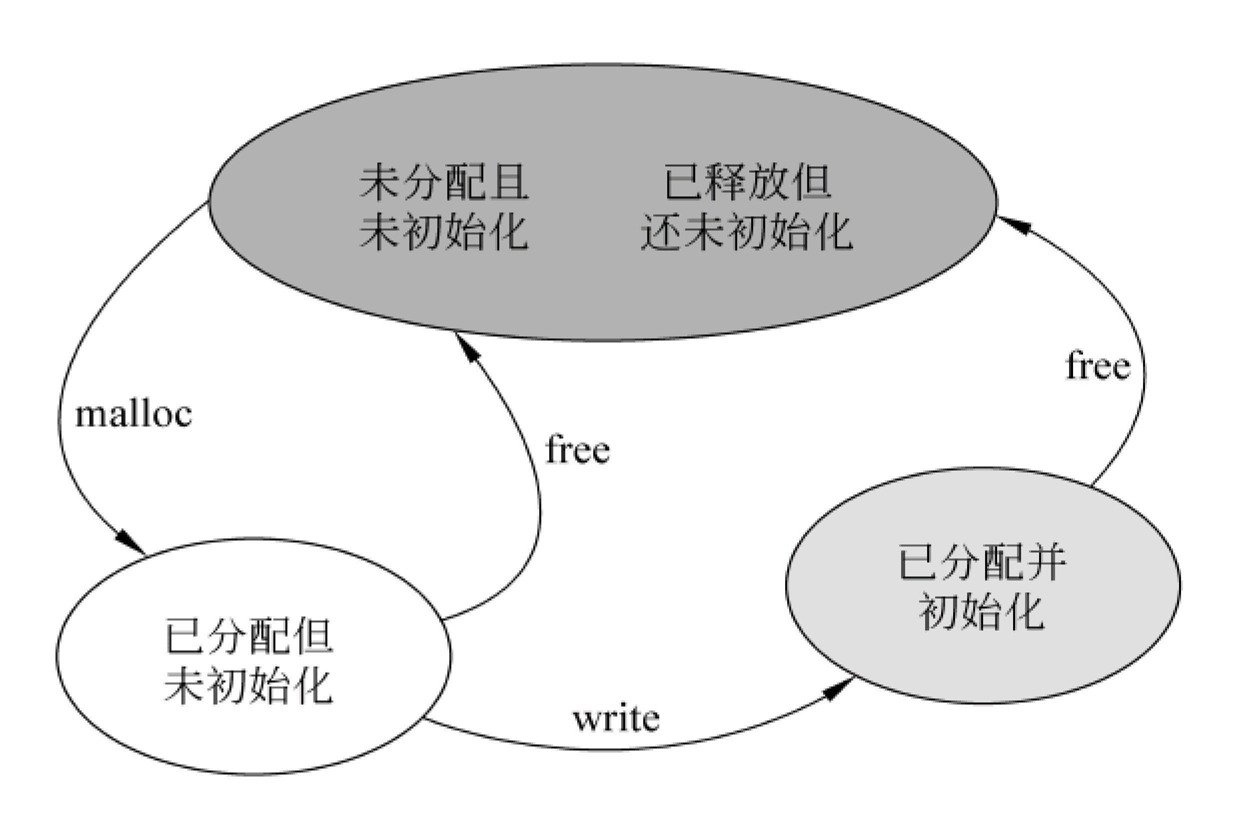


图6-7　内存访问错误图

参见本节的“1. 内存访问错误”中给出的代码，在程序第5行执行后，str2处于黄色状态。当在第7行进行读的时候，系统就会报告一个访问未初始化内存错误(uninitialized memory read)。因为只有在绿色状态下，内存才可以被合法访问。

为了检查数据越界错误(ABR和ABW)，Purify还在每个分配的内存前后插入了红色区域。这样一来，超过边界的访问指令必定落在非法区域，从而触发ABR或ABW错误报告。这里需要指出，访问未初始化内存错误UMR在某些情况下其实是合法的操作，例如内存复制。所以在分析报告时可以把UMR放到最后，或者干脆从结果中滤除。

5. 使用方法

开发人员可以用两种方法来使用Purify。首先，当遇到一个程序错误或看起来是与不适当使用内存相关的缺陷时，可以把它当作一个程序调整工具来使用。当内存被毁坏、不适当地使用或泄漏时，Purify能够识别所有不正当行为。

还有一种使用Purify的方法：像作家利用拼写检查器一样，找到并去除程序员没有意识到的错误。有规律地使用Purify可以在程序崩溃或其他明显疑问的行为发生之前，找到内存的错误使用并修复它。

中兴是怎么使用Purify的

中兴程序设计部门使用Purify的方法是，将它作为源代码中的自动步骤来控制过程。在开发人员的检入代码被允许进行测试之前，必须用Purify来进行测试。如果存在Purify错误报告，这个变更是不能继续进入到这个构架。这样可以使构架保持健康，也不至于使开发人员陷入困境中。

我们都不希望引入一个变更而导致所有测试崩溃。Purify就像维他命或预防免疫药剂，程序员可以每天使用它来保持代码的健康，如果还有问题，也可以在掌控之中。

在运行测试自动化时，使用Purify进行内存分析，需要以下步骤：

① 把 Purify 合并到测试自动化中，以便它能够在后台尽量不受干扰地运行，意味着要使用 purify/savetextdata 前缀运行程序，这与前面描述的 PureCoverage 和 Quantify 的技术类似。

② 收集运行时的信息，可能是泄漏、空指针或错误的内存使用。

③ 分析收集的所有数据，总结为一篇简单的文档，指出哪个测试或代码部分导致了问题。然后把这些信息传给开发人员。

④ 在问题解决后，重新运行 Purify 进行测试，并报告结果。

在 Windows 中，只要运行 Purify，填入需要分析的程序及参数就可。Purify 会自动插入检测代码并显示报告。报告的格式如图 6-8 所示。蓝色的图标代表一些运行的信息，例如开始和结束等。黄色是 Purify 给出的警告。通常 UMR 会作为警告列出。红色则代表严重的错误。每一种相同的错误，尤其是在循环中的，会被集中在一起显示，并且标明发生的次数，还有每个错误的详细信息，用户可以知道相应的内存地址和源代码的位置，并直接修改。另外用户还可以设置不同的过滤器，用来隐藏暂时不关心的消息。其中 MLK(memory leak，内存泄漏)，PMK(potential memory leak，潜在的内存泄漏)，ABR(array bounds read，数组边界读取)等的警告、错误和信息。这种每个缩写都表示一种应用程序中出现的内存错误。

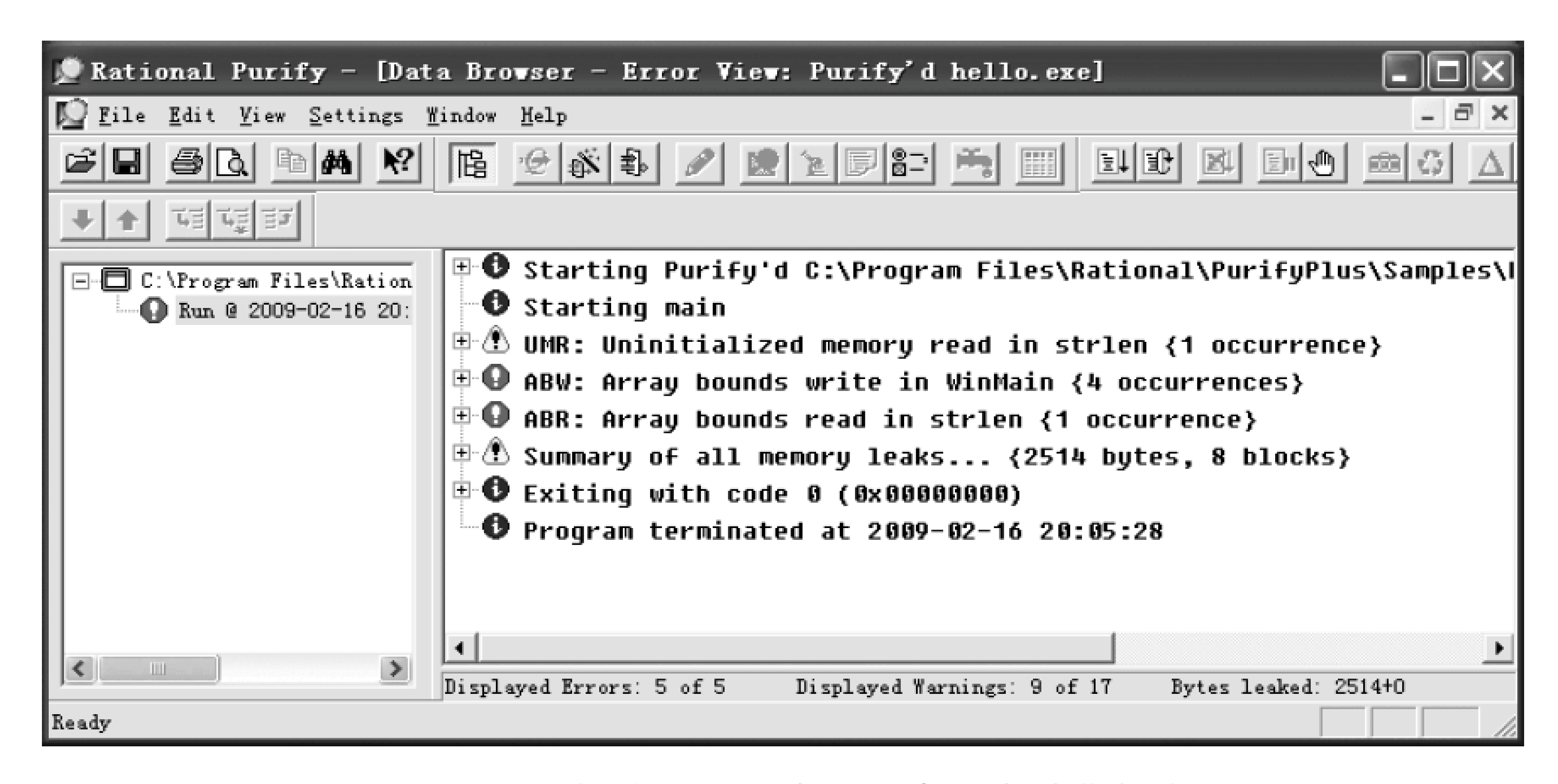


图 6-8　一个测试 C++ 应用程序的自动化报告

6. 内存分析

使用 Rational Purify 来获取对所测试的 Web 应用程序的内存分析，大体上类似于使用 PureCoverage 和 Quantify。对运行中的应用程序进行内存使用情况的"快照"可以让用户在运行的不同阶段对内存状态进行比较。这是检测内存泄漏的一个很有用的方法。

服务器应用程序常常是全天候运行。这些服务器应用程序的内存泄漏，很容易造成应用程序和系统同时崩溃，因为在运行过程中，它会不断地消耗越来越多的内存。内存泄漏对应用程序性能的影响也是巨大的。通过 Rational Purify 记录的 Java 应用程序的内存状态，可以使用户更深入地分析内存的使用状况。

Purify 在对 Java Servlet 和 JSP 进行测试时，提供的报告类型和它在测试 Java 应用程序与 Java Applets 时是一样的。图 6-9 显示了一个 Purify 记录一个接受测试的 JSP 如何调用 JavaBean 方法。

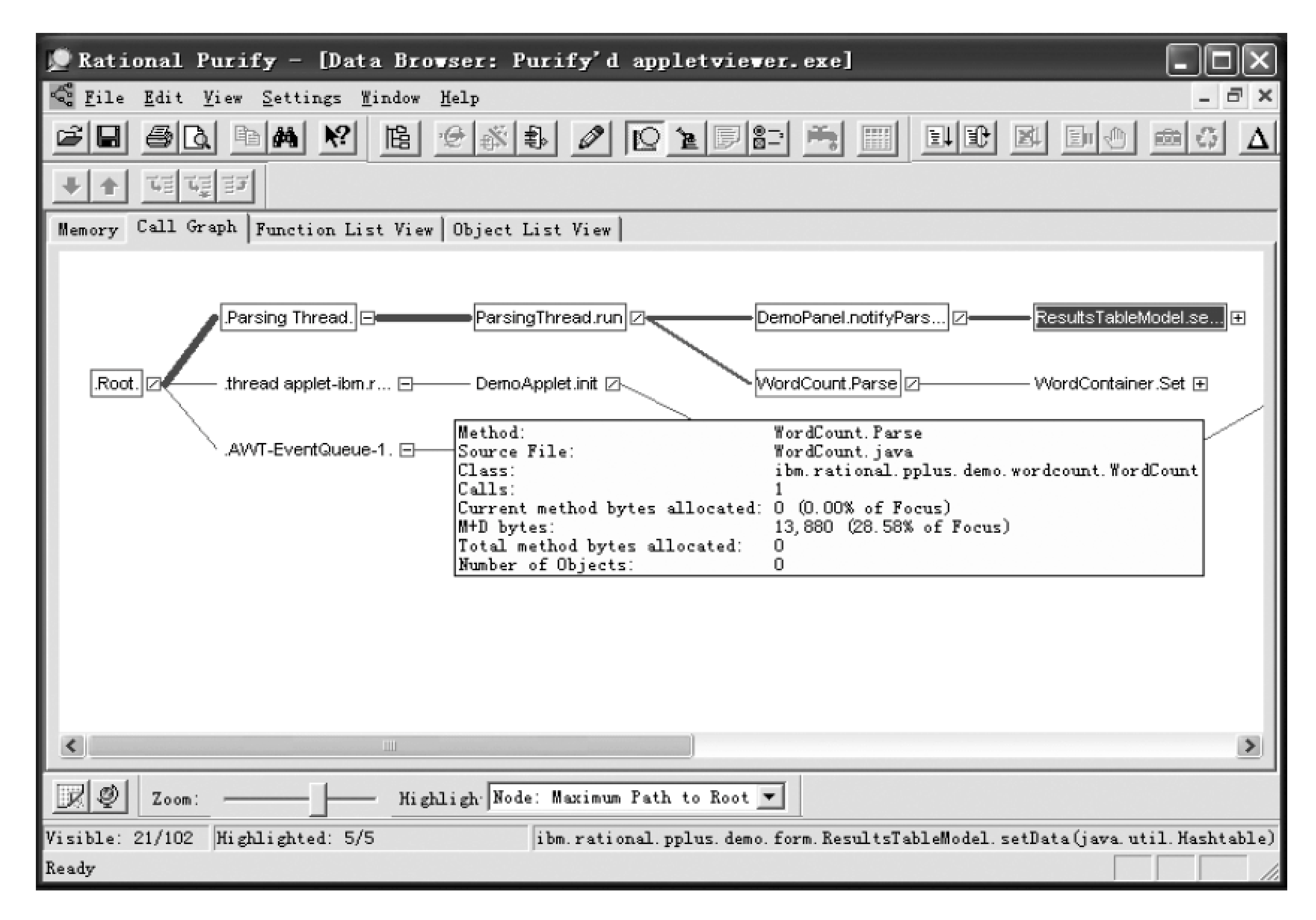
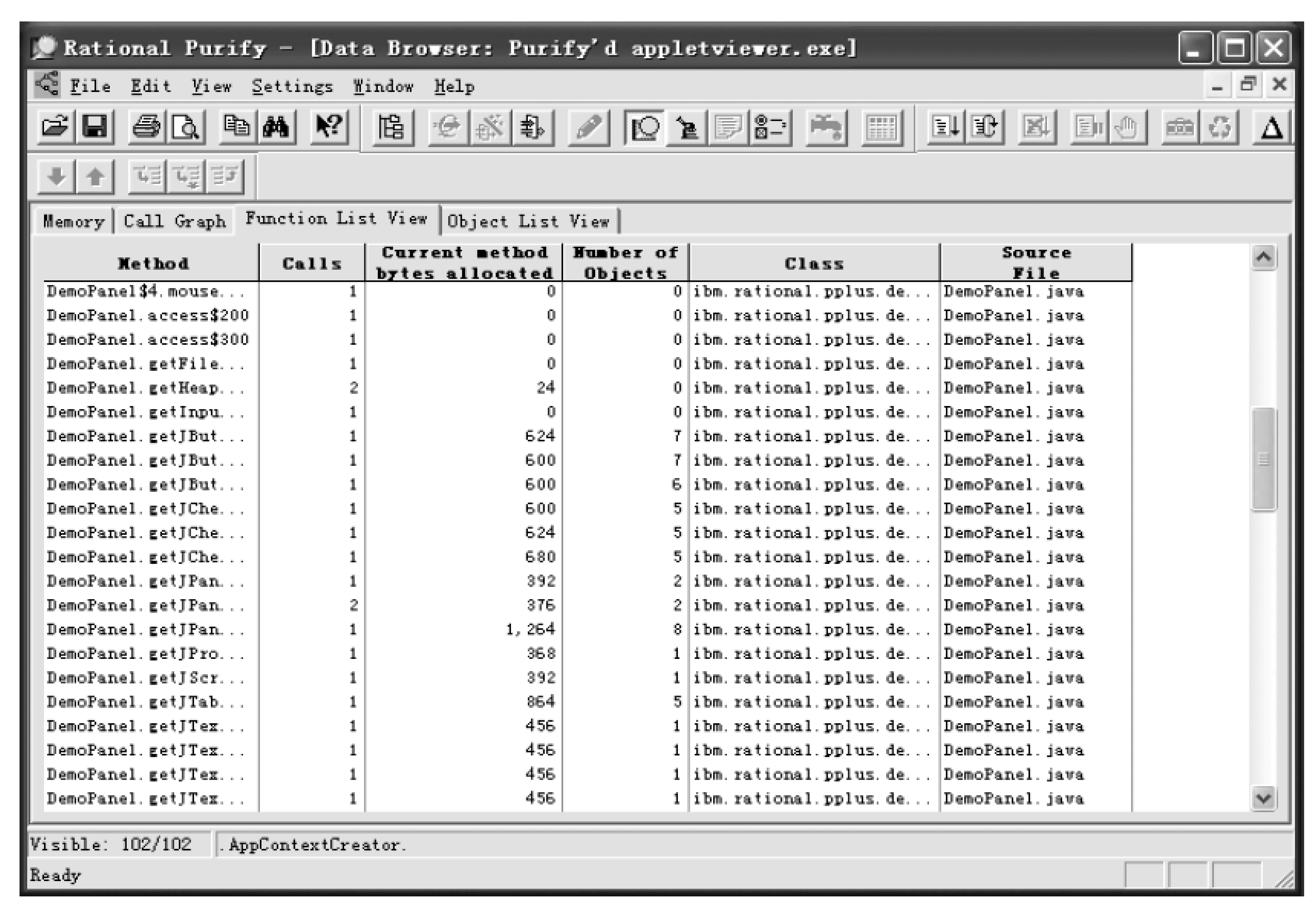


图 6-9　Demo JSP 的 Rational Purify Call Graph

如图 6-10 所示，在方法列表中提供了更多有关 JSP 会话的内存使用信息。



Method	Calls	Current method bytes allocated	Number of Objects	Class	Source File
DemoPanel$4.mouse...	1	0	0	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.access$200	1	0	0	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.access$300	1	0	0	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getFile...	1	0	0	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getHeap...	2	24	0	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getInpu...	1	0	0	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJBut...	1	624	7	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJBut...	1	600	7	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJBut...	1	600	6	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJChe...	1	600	5	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJChe...	1	624	5	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJChe...	1	680	5	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJPan...	1	392	2	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJPan...	2	376	2	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJPan...	1	1,264	8	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJPro...	1	368	1	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJScr...	1	392	1	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJTab...	1	864	5	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJTex...	1	456	1	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJTex...	1	456	1	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJTex...	1	456	1	ibm.rational.pplus.de...	DemoPanel.java
DemoPanel.getJTex...	1	456	1	ibm.rational.pplus.de...	DemoPanel.java

图 6-10　JSP 的 Rational Purify 函数列表

与 Rational Purify 引导用户发现应用程序中消耗最多执行时间的部分类似，Purify

向用户指出了应用程序的内存瓶颈所在。通过对函数列表视图中的方法进行排序，可以很容易找到那些消耗过多内存的方法，Purify 同时还提供了很多其他视图来帮助程序员做到这一点。

星际争霸历史

星际争霸诞生在 1997 年，到现在已经是 11 年了。它是全球玩家几乎都认可的最优秀的游戏，风行期有 6 年时间，其火热程度丝毫不逊色于当前流行的魔兽和 CS。直到现在，每天仍有数以百万计的玩家在线上或线下玩它。

作为最经典的游戏，其背后是高质量的支撑。星际争霸开发商暴雪公司的首席营运官 Paul Sams 表示，他们是 No. 1，有全世界最出色的游戏开发人员和测试团队，而且他们的技艺精湛。不管是程序设计还是测试，他们都是最棒的。因为他们在招聘职员时就严格把关，只招最优秀的程序员。所以他们编写的代码几乎是最完美的，再加上有严谨的项目管理，最优秀的测试团队，星际争霸的代码几乎没有任何内存错误。

正因为"暴雪十年磨一剑"的信念，无数优秀程序员夜以继日努力的成果换来了星际争霸这一最优秀的游戏软件。很多玩家认为星际争霸已经不是游戏，而是一个精美的艺术品了。

6.3.2 Quantify 组件

Quantify 是 PurifyPlus 中用来进行性能分析的组件，它帮助分析应用程序的性能瓶颈。它适用于 C/C++ 、Java 和. NET 程序。Quantify 度量并分析在程序中所花费的时间，从而识别"热点"和无效的代码。独特的 River of Time 显示可以直接让程序员找到程序中最浪费时间的区域。

1. 性能降低原因

在新 Build 的自动化测试明显变慢时，找到源代码内部发生了什么至关重要。导致变慢的原因有如下几种可能：

① 源代码中引入的变更。可能有"没用"的模块导致了运行变慢，也可能是算法的改变影响了性能。

② 测试代码的变更。增加的用来改善代码覆盖率的测试用例可能导致变慢，也可能这部分代码特别容易影响速度。

③ 测试数据的变更。大量的测试数据(或不常见的数据)可能对代码产生压力，导致瓶颈。

④ 环境的改变或者连接问题。如果网速变慢，可能是硬件需要升级了。

如图 6-11 所示，这是使用 Quantify 进行测试的 ClassicsJava 程序的一个例子。程序已经预先处理过，然后运行自动化测试。显然可以直观地看出哪些方法最耗时间，还可以看出哪些方法是调用者还是子调用者，以及这些调用花费了多长时间的信息。这样可以方便程序员检查这些信息以确定是否有可以优化的地方。

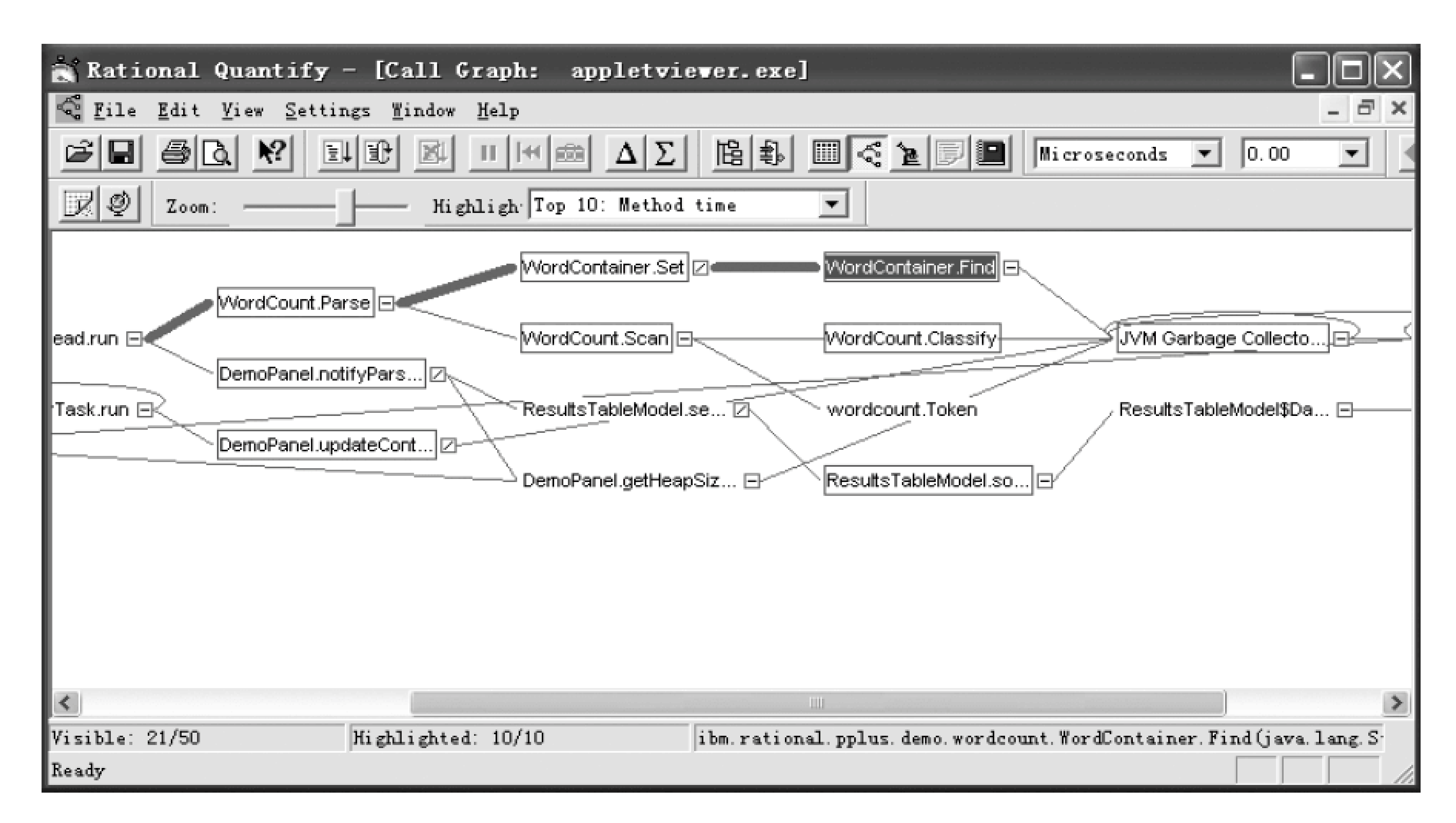


图 6-11　Quantify 生成的 Callgraph 显示出最花费时间的方法

2. 使用方法

测试人员或测试工具可以从较高的层面上告诉程序员系统性能问题所在，但是找到并修复这个潜在问题的任务将落到开发人员身上，因此这个时候 Quantify 组件就派上用场了。Quantify 可以帮助找到代码中的“热点”并标示出来，例如算法需要改良的地方，像链接列表应该是一个哈希表，或什么地方应该使用缓存，以避免不必要的重新计算等之类的问题。

Quantify 还可以用来比较两个程序的性能，一个是源程序，一个是修改后的程序，这样开发人员就可以清楚地看到代码变更时所表现的性能。因此程序员可以利用真实的代码，而不是利用猜测来对程序性能进行最优化操作。对于 C/C++ 程序，Quantify 利用不变的定时数字作为衡量标准来比较性能，这样性能数据就不会受到外界因素的影响，例如在其他测试机子上进行测试。这意味着与只利用运行时间来衡量性能相比，开发人员可以更加可靠地比较这两种趋势。

除了性能分析之外，另一种使用 Quantify 的方法是得到对程序结构和功能更好的理解。Quantify 的 River of Time 展现了一个图示，显示了程序的功能是如何调用对方的。开发人员可以利用它来跟踪程序代码，并可以查看子系统之间的相互关系。开发人员有时不希望这个子系统来调用那个子系统，只要使用 Quantify 就可以发现。如果开发人员使用 Quantify 的调用图示发现与所期望的并不匹配，那么程序就有问题，需要分析问题所在，并进行修改，使之改进得到的结果满足自己希望的目标。

使用 Quantify 进行性能分析的步骤如下：

① 决定何时收集数据。要求 Quantify 能够稳定运行一段时间，这期间系统不会出现崩溃的情况。

② 设置基准，以对比特定数量的测试的期望时间和实际时间。

③ 使用处理过的程序运行自动测试。

④ 分析收集到的数据，确定哪些测试或方法有问题。

⑤ 识别和研究这些问题，看是否可以修正，然后重复整个过程。

如果是测试人员负责测试，则把性能分析的结果发给软件开发人员，让他们找找是否有办法阻止程序变慢。同时也要寻找什么样的变更会引起性能下降。如果找到原因并进行了改进，则基准就需要调整，以便下一次测试时性能下降变得明显，以方便观察。

进行这种类型的分析，还有很多用途。例如如果自动化测试运行得很慢，不能对被测程序进行更多的测试。这个时候可以改为每天晚上进行一部分自动化测试，而不需要一次运行全部测试。这时利用 Quantify 性能分析工具合理分配测试，收集性能数据。当然需要事先确定测试计划和花费时间等。

3. Quantify 实战

(1) 使用 Quantify 运行一个程序来收集性能数据

评价程序性能的第一步是收集性能数据。对于 Visual C++，既可以直接使用 Quantify 集成的 Microsoft Developer Studio，也可以通过 Quantify 来诊断并运行一个程序。Quantify 诊断可执行文件的副本和与其相关的模块。Quantify 也可以插入其他代码来收集计数和计时性能数据。Quantify 在诊断文件的同时将显示进度。

对于 Java，既可以由 Quantify 的容器程序（使用 Run Program 对话框），也可以由命令行来运行 Java Applet、类文件或代码。在评价 Java 代码时，Quantify 将 Microsoft Virtual Machine for Java 置于一种特殊模式，Quantify 借此监测 Virtual Machine 的操作并在 Applet、类文件或代码运行的同时，直接收集计数和计时性能数据。

对于 Visual Basic，既可以直接使用 Quantify 集成的 Microsoft Visual Basic，也可以由 Quantify 来运行 Visual Basic 项目或伪码程序（Visual Basic 6.0）或 Visual Basic 本机码程序（Visual Basic 5.0 或更高版本）。在评价项目或伪码程序时，Quantify 将 Visual Basic for Applications (VBA) 解释引擎置于一种特殊模式，Quantify 借此监测该引擎的操作，并在代码运行的同时直接收集计时性能数据。对于本机程序，Quantify 将诊断该程序，然后收集计数和计时性能数据。

当 Quantify 开始评价时，它会显示 Run Summary 窗口，这样就可以监测线程和线程的活动，并检查有关此次运行的其他信息。Quantify 会在执行代码的同时记录其性能的有关数据。测试员可以随时暂停和重新开始记录数据，这样测试员就可以评价特定的代码段。也可以保留当前数据的快照，这样就可以分阶段地检查性能。

退出程序时，Quantify 将提供该程序性能的一个完整评价。由于这个数据集可能很大，所以 Quantify 在显示性能评价之前，会使用过滤器从系统库和其他模块中自动滤除不重要的数据。在分析性能数据的同时，可以或多或少地显示来自于原始数据集的数据和细节。

提示： 除了交互式地使用 Quantify 之外，也可以将 Quantify 与测试脚本、makefile 和批处理文件一同使用来进行自动测试。有关详细信息，请查阅 Quantify 联机帮助索引中的 scripts。

（2）使用 Quantify 的数据分析窗口和工具来分析性能数据

评价程序性能的第二步是分析 Quantify 收集的性能数据。退出 Quantify 为其收集数据的程序后，Quantify 会显示 Call Graph 窗口，用图形的方式来描述该程序中函数、过程或方法（此处总称为函数）的调用结构和性能。默认情况下，调用图会根据函数＋子代（F＋D）时间来显示当前数据集中的前 20 个函数。Quantify 的结果实际上并未包括评价过程自身占用的时间。所看到的数字是程序在未利用 Quantify 的情况下所花费的时间。

调用图还突出显示了占用时间最多的路径，较粗的线条表明路径占用的时间较多。也可以根据不同的标准，包括性能、调用关系以及可能产生瓶颈的原因，来突出显示其他函数。还可以显示另外的函数、隐藏函数，并来回移动函数，以便更清楚地查看调用图。

可以使用 Quantify 的其他数据分析窗口来进一步检查程序的性能。要对当前数据集中的所有函数进行复审，并根据不同标准对其分类，请使用 Function List 窗口。Function Detail 窗口以表格加图形的形式显示了特定函数的数据，以及有关其调用方和后代的数据。如果在运行程序时，调试数据是可用的，并且测试员是在代码行一级上评测函数，那么测试员就可以使用 Annotated Source 窗口，逐行地分析特定函数的性能。

Quantify 提供了几种途径来缩减较大的数据集，从而只显示感兴趣的数据。例如，可以指定过滤器根据模块、模式（如名称中有 CWnd 的函数）或评测类型（如所有等待和阻塞的函数）来滤除函数。也可以集中在一个特定的子树上。

通过合并各次独立的运行过程来创建一个新的数据集，可以很容易地跨越各次运行来分析程序的性能。

（3）再次运行该程序并使用 Quantify 的比较运行工具来查找性能变化

评价程序性能的第三步也是最终步骤，是比较两次运行的性能数据，查看改动代码是使性能提高了还是降低了。改动代码后，可以重新运行更新过的程序并将新结果同前一次运行作比较。Diff Call Graph 用绿色突出显示性能的提高，用红色表示性能的降低，这样可帮助迅速地查明性能的改变。Diff Function List 除了显示两次运行的原始数据之外，还显示两次运行之间的差别。

使用 Navigator 窗口来跟踪执行的各次运行过程。测试员可以将性能数据保存为一个 Quantify 数据文件（.qfy），用于进一步的分析或同其他 Quantify 用户共享。测试员也可以将数据保存为一个由制表符分隔的 ASCII 文本文件（.txt），从而能够在 Quantify 之外（如在测试脚本中或在 Microsoft Excel 中）使用。还可以直接将数据由 Function List 窗口复制到 Excel 中使用。

6.3.3 PureCoverage 组件

前面已经介绍了怎么使用单元测试代码，但测试员怎么评价他的测试呢？他怎么发现没被测试的代码呢？怎么衡量测试的完整性？所有这些问题的答案就是代码覆盖率分析。代码覆盖率分析可以告诉测试员当测试进行时，哪些代码执行过了等方面的问题。

1. 代码覆盖率分析

代码覆盖率分析就是找到定位没用的或不执行的代码的过程。没用的代码不会存在什么问题，但是它们会影响程序的可读性；不执行的代码则可能是未来 Bug 的所在。所以把它们从程序中找出来并进行修改或删除是大有裨益的。

图 6-12 是使用 PureCoverage 来测试 ClassicsJava 的例子。ClassicsJava 是 IBM Rational Functional Tester 中的一个示例程序，功能是对数据库中的音乐 CD 进行排序。在图 6-12 中可以直观地看到，被覆盖的方法加上了 Hit 标签，没有覆盖的则加上 Missed 标签。

Module View | File View

Coverage Item	Calls	Methods Missed	Methods Hit	% Methods Hit	Lines Missed	Lines Hit	% Lines Hit
ClassicsJava.java	17	9	13	59.09			
ClassicsJava.<clinit>()	1		hit				
ClassicsJava.ClassicsJava()	3		hit				
ClassicsJava.PopulateTree(java.util.Vec..	1		hit				
ClassicsJava.access$000(ClassicsJava..	1		hit				
ClassicsJava.class$(java.lang.String)	1		hit				
ClassicsJava.fillImageRecordset()	1		hit				
ClassicsJava.floatToDollars(float)	3		hit				
ClassicsJava.getCustomerNames()	0	missed					
ClassicsJava.main(java.lang.String [])	1		hit				
ClassicsJava.makeAboutFrame()	0	missed					
ClassicsJava.makeAdminLogon(java.la..	0	missed					
ClassicsJava.makeClearDataForm()	0	missed					
ClassicsJava.makeCustomerAdminForm..	0	missed					
ClassicsJava.makeMenu()	1		hit				
ClassicsJava.makeOrderAdminForm()	0	missed					
ClassicsJava.makeOrderForm(Customer)	1		hit				
ClassicsJava.makeOrderLogon(java.util..	1		hit				
ClassicsJava.makeOrderStatusForm(Cu..	0	missed					

图 6-12 PureCoverage 显示自动化测试程序的 Hit 和 Missed

覆盖率分析主要有下面的几个过程：通过测试程序组找到不执行的程序段；添加额外测试程序组，以便增加代码覆盖率；决定代码覆盖率的定量测度，它也是程序质量的间接测度。

代码覆盖率分析不能找出程序的逻辑错误。考虑一下下面的代码：

```
10: rc=call_to_xx ();
11: if (rc==ERROR_FATAL)
12:    exit(2);      /* exit with error code 2 */
13: else
14:    /* continue on */
```

当测试程序段运行到第 11 行时，第 11 行始终都不能为真。call_to_xx 返回了另外的一个错误如 ERROR_HANDLE，除非加入这种错误的处理方式的代码。

PureCoverage 作为代码覆盖测试工具，只能显示已经存在的代码的覆盖率。利用 PureCoverage，可以确保测试整个程序。毕竟没有进行测试的代码的质量是未知的。因为一般测试并不报告它的错误或健康状态。没有测试到的错误将会“逃离”进入释放，找到并修复它们的成本将十分昂贵。在发布版本之前识别测试漏洞，填充并找到这些错误，成本会低得多。

PureCoverage 通过记录执行过的代码，生成代码覆盖分析报告，其代码覆盖分析可以详细到语句级，举个例子来说，如果有一个方法有 100 个代码行，在测试进行时，只有 75 行真正运行了，这个方法的语句覆盖率就是 75%。

PureCoverage 主要使用目标码插入 OCI(object code insertion)技术来实现检测测试覆盖程度，不过源代码插入 SCI(source code instrumentation)和执行码替换 ECI(executable code interception)都需要源代码或编译环境的支持，并且会引起程序运行缓慢和系统资源占用过多的问题。但是瑕不掩瑜，PureCoverage 凭借着与 Visual Studio 集成开发环境的无缝连接，依然是单元测试工具的首选利器。此外，很重要的一点是，Purify 只在运行测试的过程中对部分的程序进行查找缺陷。如果测试员在测试覆盖范围中有漏洞，那么在 Purify 错误检验中仍然会有漏洞。这是使用 PureCoverage 另一个很好的理由。

2. 冗余度测试

覆盖率分析也可以帮助找到测试用例是否有冗余：测试在代码的同一路径下反复运行，导致了不必要的时间延迟。另外它也可以帮助确认测试数据。例如，一个新引进的代码变更，需要运行自动化测试以进行回归确认。这时就需要检查覆盖率分析的数据，以帮助确定哪些自动化测试的子集需要运行，这样就可以在更短的时间内验证新的代码。

为了深入到特定的方法内部，可以使用 PureCoverage 的 Annotated Source 特性，为了使用这个特性，PureCoverage 必须能够访问源代码。一般是集成到开发人员的工作环境和软件，这样 PureCoverage 就可以很容易访问所有的源代码路径。图 6-13 显示出 Annotated Source 视图。正如同图 6-12 显示的那样，这个方法被遗漏了，因此代码行用红色字体显示，图中是第 828～855 行。

Line Coverage	Line Number	
1	822	}
	823	public String[] getCustomerNames()
	824	{
	825	try
	826	{
	827	
0	828	String query = URLEncoder.encode("SELECT NAME FROM CUSTOMERS","US-ASCI
	829	// URL url = new URL(getCodeBase()+"cust.asp?custQuery="+query);
0	830	URL url = new URL("cust.asp?custQuery="+query);
0	831	URLConnection connection = url.openConnection();
	832	
	833	//DataInputStream in = new DataInputStream(connection.getInputStream()
0	834	BufferedReader in = new BufferedReader(new InputStreamReader(connectio
	835	
0	836	int count = 0;
0	837	Product recordset[] = new Product[50];
0	838	String line = in.readLine();
0	839	StringTokenizer st = new StringTokenizer (line, "**");
0	840	String[] custNames = new String[100];
0	841	while (st.hasMoreTokens())
	842	{
0	843	custNames[count]=(st.nextToken());
0	844	count++;
	845	//System.out.println(count);
	846	}
0	847	return custNames;
	848	
	849	}
	850	
	851	
	852	
0	853	catch (Exception e){
0	854	System.out.println(e.getClass().getName()+": "+e.getMessage());
0	855	return null;

图 6-13 PureCoverage 显示方法 getCustomerNames() 的注释源视图

3. PureCoverage 使用步骤

进行代码覆盖率分析的主要步骤为：

① 自动化运行由 PureCoverage 处理过的应用程序。

② 收集覆盖率数据。

③ 分析数据，找到哪些代码没有运行。

④ 增加测试用例，尽可能地覆盖没有运行的代码。

⑤ 运行增加了测试用例的自动化测试。

⑥ 确认增加的用例改善了覆盖率。

4. PureCoverage 实战

PureCoverage 有两种运行界面，一种是应用程序界面，可以脱离开发环境独立运行；另一种是嵌入式界面，可以集成到 Visual Studio 的集成开发环境中，不过两种运行方式的基本功能是相同的。为了让读者对 PureCoverage 有一个感性的认识，这里以 PureCoverage 自带的一个例子 hello. exe 作说明。

首先运行 PureCoverage，然后选择 File 菜单的 Run 命令，在弹出的 Run Program 对话框中选择 hello. exe 程序，如图 6-14 所示。

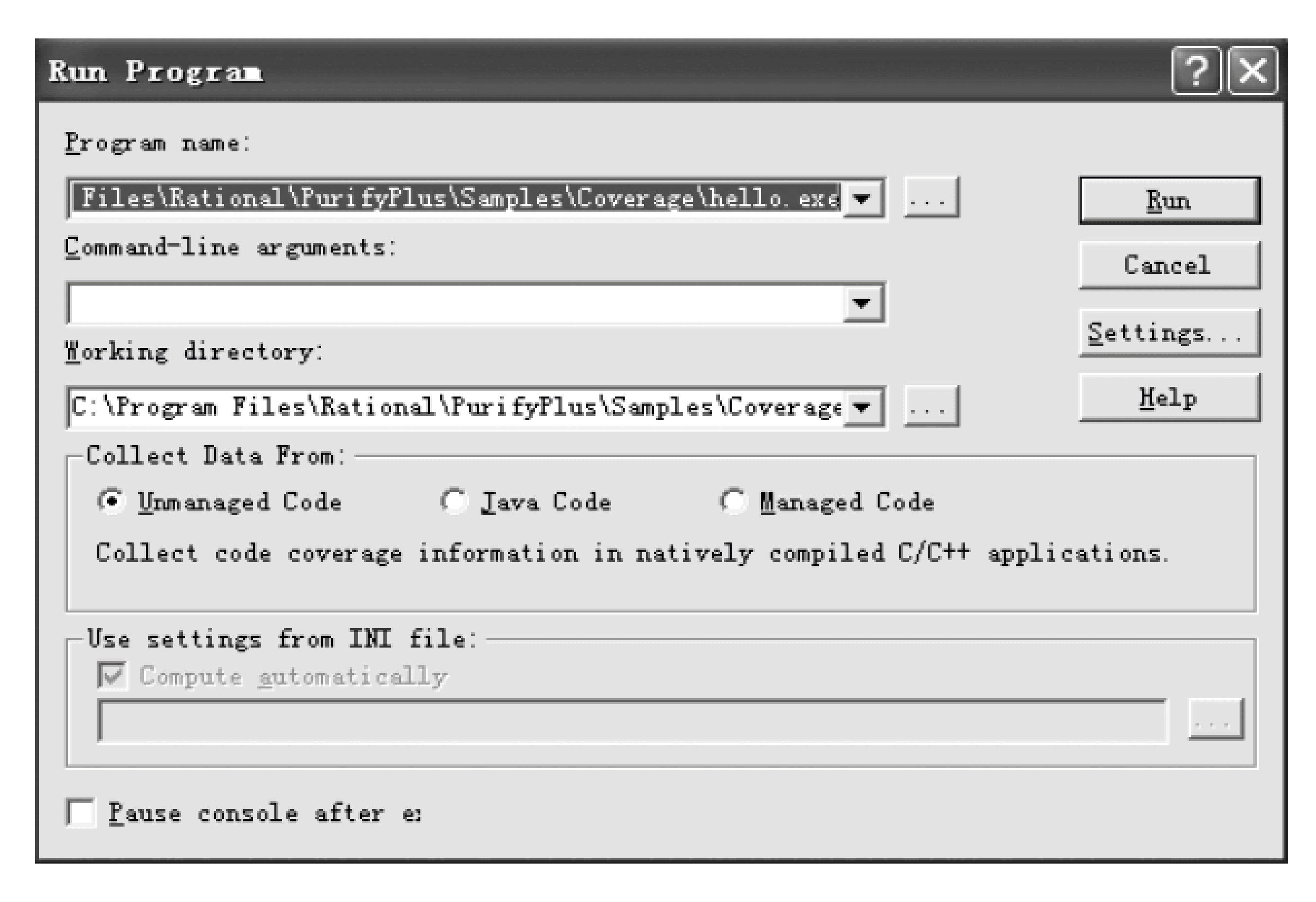


图 6-14　选择程序

由于 hello. exe 是使用 Visual C++ 6.0 编写的 Win32 应用程序，所以在 Collect Data From 选项处选择 Unmanaged Code(非托管代码)，单击 Run 按钮就开始运行 hello. exe。PureCoverage 首先分析 hello. exe 装载的模块，然后运行 hello. exe，出现一个标题是“Hello，World”的消息框，这是 hello. exe 程序的提示，要用户选择是不是看看当前的时间，可以单击“确定”按钮看结果。hello. exe 弹出另一个消息框显示当前时间，单击“确定”按钮结束程序的运行，此时 PureCoverage 已经统计出了结果，如图 6-15 所示。

这个视图显示了 hello. c 有两个函数 DisplayLocalTime() 和 WinMain()，在刚才那次测试运行中被调用的次数和代码行覆盖率。用鼠标双击函数名可以将视图切换到代码

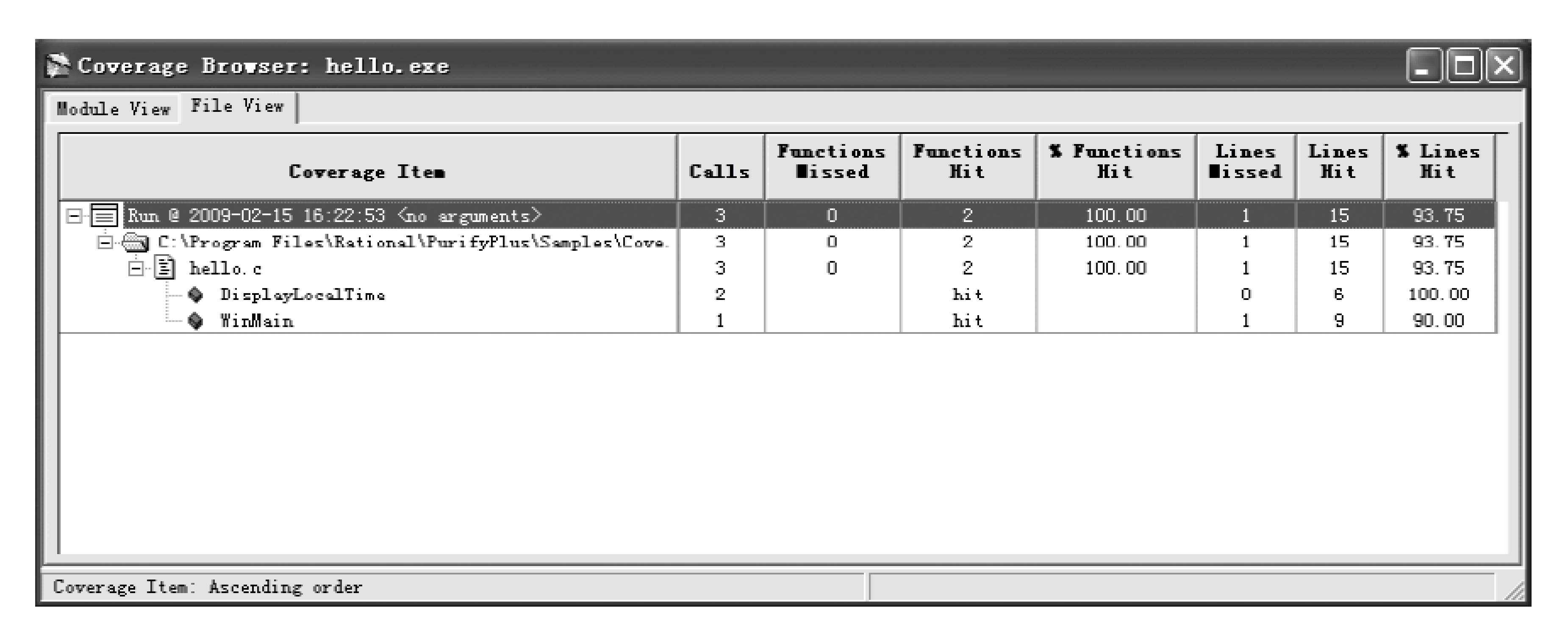


图 6-15　运行结果的 Module View 视图

窗口，如图 6-16 所示。

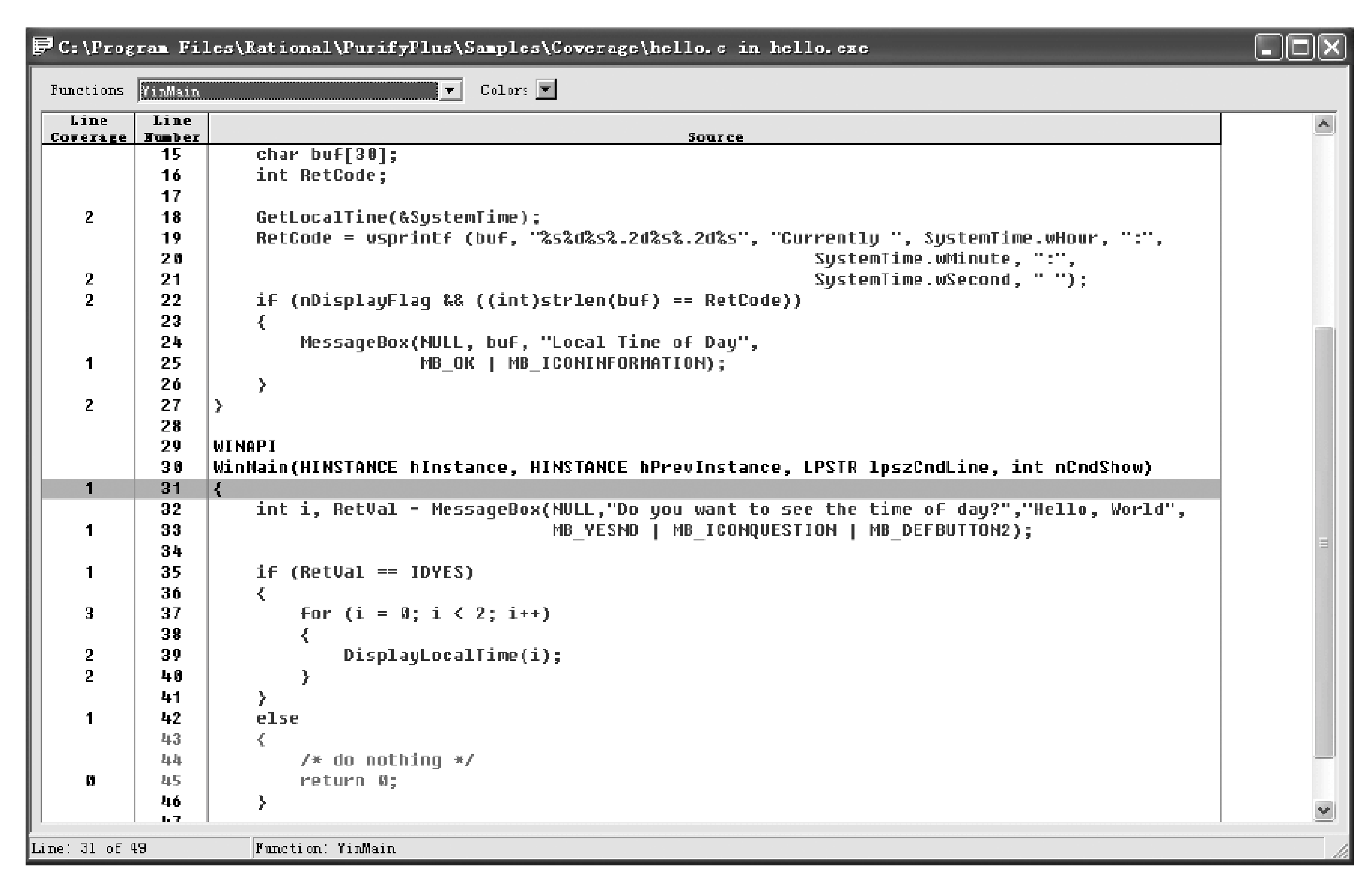


图 6-16　运行结果的 Source Code 视图

代码视图中被标记为蓝色的代码是刚才测试走过的代码，被标记为红色的代码表示这次测试没有走过的代码。如果想了解每个函数的具体情况，可以点击工具栏的 Function List 图标，图 6-17 就是本例中的函数列表，包括函数被调用的次数、总的代码行数、被测试过的行数以及代码覆盖率。

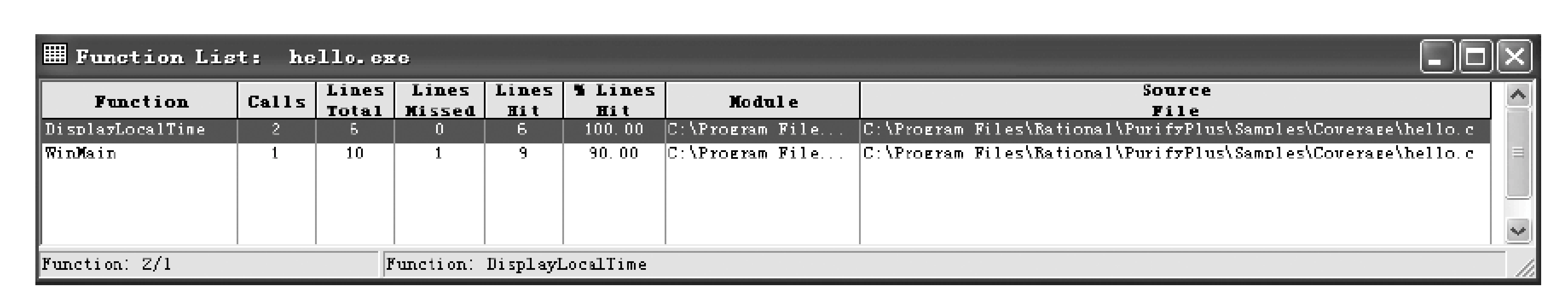


图 6-17　Function List 显示

有时候一次测试通常不能覆盖所有的代码，以上面的 hello.exe 为例，WinMain()函数的一个分支就没有走到，PureCoverage 充分考虑到了这种情况，它能够记录每一次测试结果，并提供了对结果进行分析、对比和合并的功能。下面再运行一次 hello.exe，这次选择不看当前时间，使程序走过 WinMain()函数的另一个流程。此时 PureCoverage 右边的 Navigator 窗口就记录下了对 hello.exe 的两次测试结果，如图 6-18 所示。

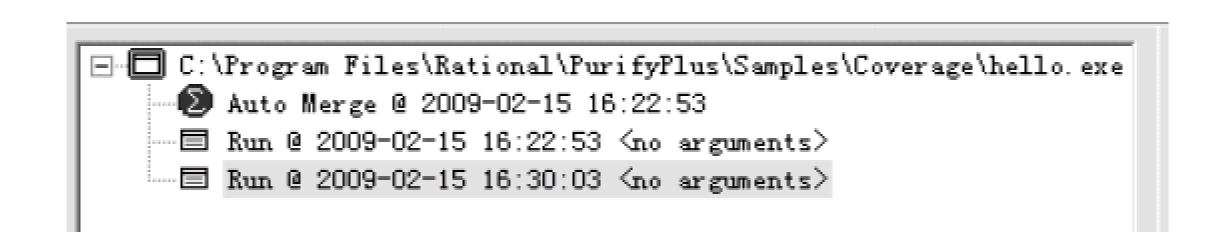


图 6-18　查看 Navigator 窗口

在 Navigator 窗口上用鼠标双击第二次测试的结果条目，就可以在右边看到这次测试的详细信息，如图 6-19 所示，这次测试没有调用 DisplayLocalTime()函数，只走过了 WinMain()函数的 else 分支。

C:\Program Files\Rational\PurifyPlus\Samples\Coverage\hello.c in hello.exe

Functions: WinMain　Color:

Line Coverage	Line Number	Source
	23	`{`
	24	`MessageBox(NULL, buf, "Local Time of Day",`
0	25	`MB_OK | MB_ICONINFORMATION);`
	26	`}`
0	27	`}`
	28	
	29	`WINAPI`
	30	`WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpszCmdLine, int nCmdShow)`
1	31	`{`
	32	`int i, RetVal = MessageBox(NULL,"Do you want to see the time of day?","Hello, World",`
1	33	`MB_YESNO | MB_ICONQUESTION | MB_DEFBUTTON2);`
	34	
1	35	`if (RetVal == IDYES)`
	36	`{`
0	37	`for (i = 0; i < 2; i++)`
	38	`{`
0	39	`DisplayLocalTime(i);`
0	40	`}`
	41	`}`

Line: 31 of 49　Function: WinMain

Function List:　hello.exe

Function	Calls	Lines Total	Lines Missed	Lines Hit	% Lines Hit	Module	Source File
DisplayLocalTime	0	6	6	0	0.00	C:\Program File...	C:\Program Files\Rational\PurifyPlus\Samples\Coverage\hello.c
WinMain	1	10	5	5	50.00	C:\Program File...	C:\Program Files\Rational\PurifyPlus\Samples\Coverage\hello.c

Function: 2/1　Function: WinMain

图 6-19　第二次测试结果

此时在 Navigator 窗口中可以看到两个测试结果和一个自动合并结果，每测试一次，PureCoverage 就会自动将当前测试的结果合并到最上边的 Auto Merge 结果上。用户可以选择某几个测试结果合并成一个结果，例如在某种特殊条件下的几次测试就可以合并到一个结果中，以便集中统计在这种特殊条件下的测试结果。合并操作非常简单，在需要合并的测试结果上单击鼠标右键，在弹出的菜单中选择 Merge Runs 菜单，然后在弹出的结果列表中选择另一个测试结果就可以将两者合并成一个结果。现在用鼠标双击 Navigator 窗口的 Auto Merge 结果查看两次测试的合并结果。图 6-20 显示了两次测试的合并结果，两个函数各被调用两次，代码覆盖率都是 100%。

生成一个比较结果也很简单，在 Navigator 窗口中选择一个测试结果，单击鼠标右键，接着在弹出的菜单中选择 Compare Runs，最后再选择另一个结果进行比较就可以了，PureCoverage 会自动生成一个比较结果，用鼠标双击这个比较结果，右边就会显示详细

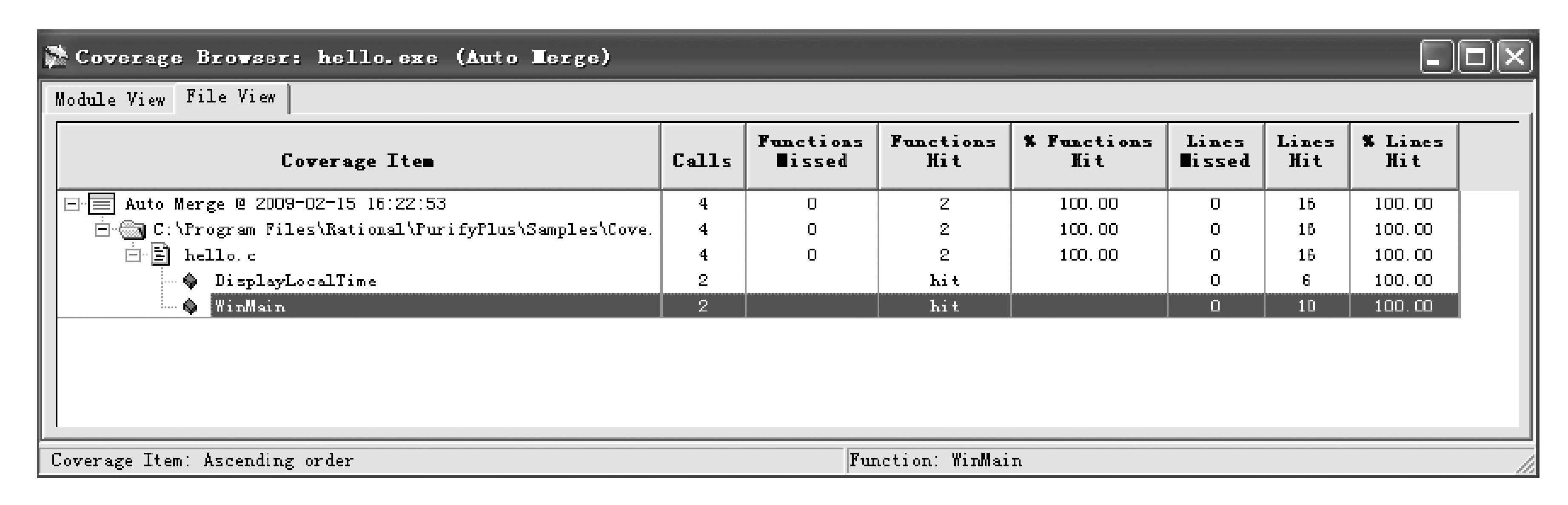


图 6-20　合并结果查看

的内容，如图 6-21 所示，Base 是原来的结果，New 是新结果。

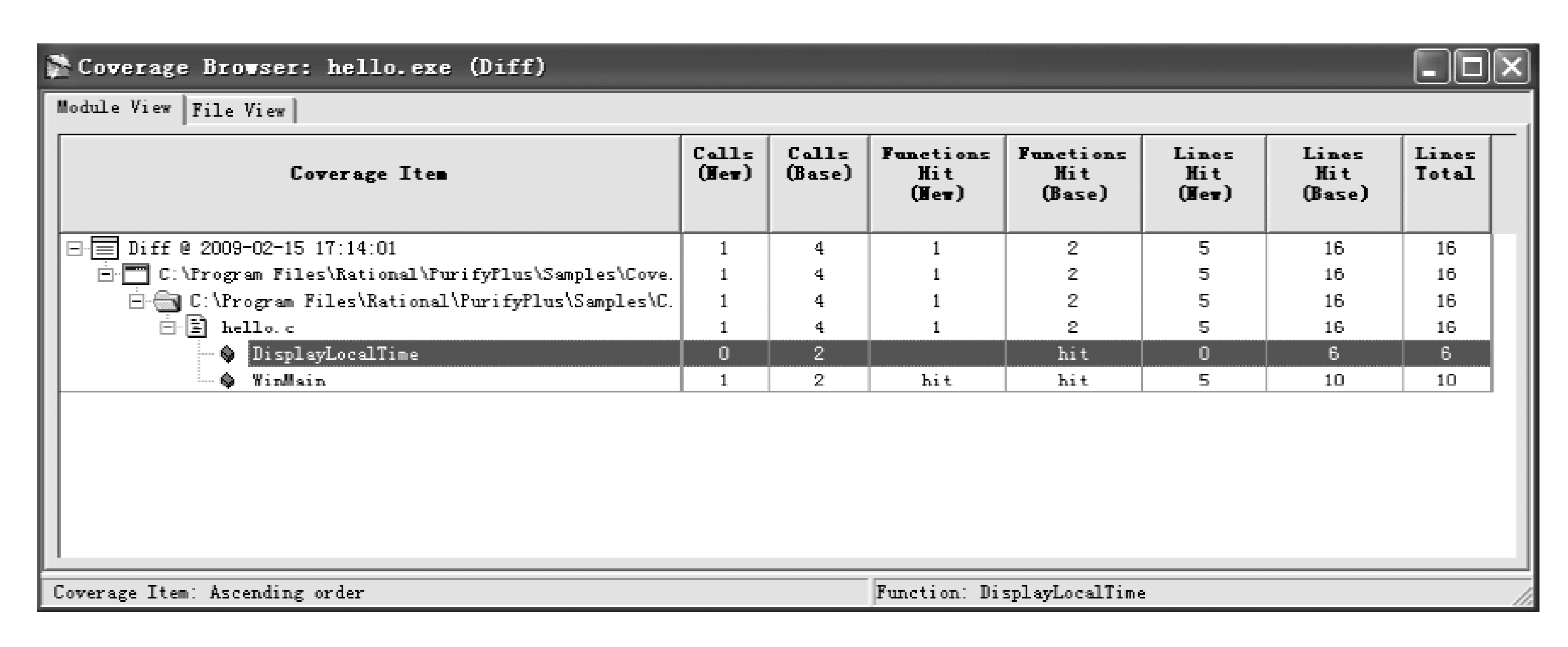


图 6-21　查看结果比较

通过上面的介绍，相信大家对 PureCoverage 已经有了初步的了解，PureCoverage 的用法非常简单，本文只是用一个简单的例子介绍了它的基本功能。使用 PureCoverage 配合 Visual Studio 开发工具，能够极大地提高软件开发的质量。

6.4　小　　结

单元测试会让开发工作变得更加轻松，让程序员对自己的代码更加自信。无论是大型项目还是小型项目，无论是时间紧迫的项目还是时间宽裕的项目，只要代码不是一次写完永不改动，编写单元测试就成为编写代码不可缺少的一部分。

内存泄漏是个复杂的问题，即使是 Java 和. NET 这样有 Gabarge Collection 机制的环境，也存在着泄漏的可能，例如隐式内存泄漏。由于篇幅的限制，本文只能对这个主题做一些粗浅的研究。其他问题，例如多模块下的泄漏检测，如何在程序运行时对内存使用情况进行分析等，读者可以查阅更多的资料。

当使用 C/C++ 进行开发时，采用良好的一致的编程规范，是防止内存问题第一项也是最重要的措施。在此前提下，IBM Rational Purify 作为一种运行时分析软件可以很好地帮助发现内存问题，将成为软件自动化测试中的一个重要组成部分。

习题与思考

1. 什么是单元测试?
2. 单元测试有哪些优点?
3. 描述单元测试的范畴。
4. IBM 公司的单元测试工具 Rational PurifyPlus 包括哪些套件?
5. 内存错误难以发现的主要原因是什么?
6. 为什么内存使用错误比内存访问错误更难以发现?
7. 以发生的方式来分类,内存泄漏可以分为哪几类?
8. 简述 Quantify 进行性能分析的步骤。
9. 覆盖率分析包含哪些过程?
10. 简述 PureCoverage 的使用步骤。

第3部分

集成测试

很多 Bug 不是在程序本身找到的，而是在其与其他程序交互时找到的。

——微软亚洲工程院软件测试部经理陈天

这句话告诉程序员 Bug 还存在于程序交互中，表明了集成测试的重要性。根据 RUP 理论，执行集成测试，是为了确保在实施模型中的组件集成起来执行用例时，这些组件能够正常运行。测试对象是实施模型中的一个包或一组包。要集成的包通常来自于不同的开发组织。集成测试将揭示包接口规约中不够完整或错误的地方。

通过这一部分的学习，读者可以获得集成测试的相关知识，了解组件测试和运行时分析定义和解决方案。

第7章 组件测试与运行时分析

组件在中国

20 世纪 80 年代中期，国家为了追赶世界科技先进水平，将软件技术研究纳入 863 计划项目，并给予了极大的支持。著名的青鸟工程，就是政府投入多、持续时间长、规模大、涉及人员和单位广的国家级项目。北京大学杨芙清院士联合了全国 22 所高校和科研单位，领导并主持开发此项目。

在开发过程中，杨芙清院士第一次在国内提出了组件的概念，并实现了基于组件/构架模式、采用集成组装方式的软件工业化生产技术。这在国际上也是最早的，卡耐基梅隆大学直到 2001 年才提出组件的定义。基于这些思想，杨芙清院士领导研制了支持组件技术的应用系统集成(组装)环境——青鸟软件生产线系统，从管理、组织机制、技术和系统等方面为软件企业提供了整体解决方案。

青鸟工程研制成功以后，引起了微软总裁比尔·盖茨的注意，1993 年比尔·盖茨访华专门参观了青鸟软件工程生产线，还在工作台上亲自操作了一番。他十分惊讶，感叹中国的软件业发展已经达到了较高的水平。

从 1990 年初期开始，青鸟工程的成果已经在众多的软件企业得到了应用和推广。截止 1998 年，青鸟软件生产线系统、领域开发平台、应用系统开发初步统计直接经济效益超过 9190 万元。1999 年至今以软件组件库为例，公共组件库组件总数超过 29 000 个，自主组件总数超过 4000 个，组件企业用户超过 670 家，形成产值 21 232 万元。神州数码用了组件技术以后，其承担的安徽国税项目代码复用率达到 76%，每年节约成本 400 万元。联想亚信采用组件技术节约成本 2250 万元。

由此可见，组件技术对我国软件工程技术的研究、实践和发展，促进我国软件产业的技术提升作出了重要贡献。

在软件开发过程中，完成模块的开发之后，首先要做的工作就是对各个模块进行单元测试，然后就是将通过测试的模块按照软件的设计文档组合成不同的组件。

基于组件的软件设计是现阶段软件开发方法的主流。这一方法的特点是能够实现软件的大粒度复用、降低软件开发和维护成本。用这种方法开发的软件系统的特点是，不同组件用不同的编程语言开发，部分组件需要运行在不同的操作平台上，在地理位置上相距很远，一些组件是自主开发的，而另一些组件是由第三方开发或是商业组件，许多组件的源码并不公开等。这样基于组件的软件开发就存在一个所谓的组件信任问题，解决这一

问题的有效方法就是进行软件集成测试。对组件开发者而言，通过对组件的测试，提供组件的质量证明，增加组件使用者使用该组件的信心；对组件使用者而言，组件测试是保证各组件间能正常有效地配合工作的集成测试，是保证能正确完成系统要求的系统测试。

学习本章读者要理解组件技术和运行时分析技术，对组件的特点有所了解，同时重点掌握组件测试的方法和特点，最后掌握测试工具 IBM Rational Test Realtime 的使用。

7.1 组件技术

Aleln 的调查表明，截止 2003 年，70％以上的软件系统大部分依赖于基于组件的软件。一个基于组件的软件系统通常由一系列独立的组件集成。其中，有些由组织内部开发，有些可能是由第三方来开发，此时源代码可能对开发者来说是不可见的。组件可以用不同的编程语言编写，在不同的操作平台上运行，同时分散在不同的地理位置上。基于组件的软件特征，可使得软件升级速度加快，同时确保最终产品的质量。

7.1.1 组件的产生

在计算机软件发展的早期，一个应用系统往往是一个单独的应用程序，应用越复杂，程序就越庞大，系统开发的难度也就越大，而且一旦系统的某个版本完成以后，在下一个版本出来之前，应用程序不会再有所改变。但对于庞大的程序来讲，更新版本的周期很长，在两个版本之间，如果由于操作系统发生了变化，或者硬件平台有了变化，应用系统就很难适应这种变化，所以这类单体应用程序已经不能满足计算机软件的发展需要。

从软件模型角度来考虑，一个很自然的想法就是把一个庞大的应用程序分成很多个模块，每一个模块的功能保持一定独立性，在协同工作时，通过相互之间接口调用来完成实际的任务。我们把每一个这样的模块称为组件，一个设计良好的应用系统往往被划分为许多组件，这些组件可以单独开发，单独编译，甚至单独调试和测试，以获取更好的维护性能和代码复用率。如果有必要，还可以从第三方购买所需要的商业组件（COTS 组件）。当所有的组件都已经完成后，把它们组合在一起便得到了一个完整的应用系统。当系统外界的软硬件环境发生变化或用户的需求有所更改时，并不需要对系统中所有的组件修改，只需要完善受影响的组件，然后重新组合，就可以得到升级的软件。

名词解释：COTS

COTS 的全称是 Commercial-Off-The-Shelf，即商务现货供应，指使用“不再做修理或改进”的模式出售的商务产品，这种产品设计的原则就是安装简便，并且可以在现有系统部件的条件下运行。COTS 组件是指商用的通用组件，它是为一个市场领域而设计的，其价格也是市场驱动的。由于市场的竞争性和开放性，使它的产品性能不断增强，品质和技术不断提高。计算机用户所要买的软件几乎都可以是 COTS 类别的产品：操作系统、Office 产品组合、字处理以及电子邮件程序就是其中的几个例子。COTS 软件的最大优点，就是它的大量生产以及相对的低成本。

由于规模经济和增加的互操作性，商用现成构件 COTS 能够使整个系统的成本降低。COTS 开发者致力于开发高级技术来生产横向的低价位的构造模块，这些模块能够在不同的纵向解决方案中使用。例如，一个设计巧妙的商用组件能够用于电信交换、语音邮件、交互语音响应（IVR）、统一消息和通信等方面。这使电信设备制造商（TEMs）、原始设备制造商（OEMs）和其他解决方案提供商能够集中精力组合多种组件并且增加特殊的纵向业务，从而减少解决方案开发时间，降低成本。COTS 组件因为是构建在现成可得的、使用广泛的成熟部件的基础上，所以大大降低了风险。

7.1.2 组件的定义

组件技术是近年来发展起来的一种优秀的软件复用技术。基于组件的软件工程（component-based software engineering，CBSE）是一种软件开发新范型。它是在一定的组件模型的支持下，复用组件库中的一个或多个组件软件，通过组合手段，高效率、高质量地构造应用软件系统的过程。

关于什么是组件，目前还没有统一的说法，有几种关于组件的定义。在 RUP 中，术语“组件”主要用来表示系统的封装部分，理论上是系统中相当重要的、几乎独立的可替换部分，它在明确定义的体系结构环境中实现明确的功能。对此，UML 定义组件如下：

封装了内容的系统模块化部分，其表现形式在环境中是可替换的。组件在提供的接口和必需的接口方面定义自己的行为。这样，组件充当一个类型，其一致性由这些提供的和必需的接口定义，包含它们静态和动态的语义。

在 RUP 中，术语“组件”的使用要比 UML 的定义广泛。程序员并不是仅将组件定义为具有诸如模块性、可部署性和可替换性之类的特征，而是建议组件应该具有这些特征。

从以上定义，可以看出组件是系统中一个有价值的、近乎独立的可替换部分，在这个系统的上下文环境中实现清晰的功能。系统应该具有良好定义的结构，要符合一组接口标准，并提供接口的物理实现。

由于软件由不同的组件构成，组件间联系是通过接口的消息传递来实现。因此接口是组件的核心部分，它只描述对象所提供的公共服务，而对象的内部状态不应该通过公共接口看到。接口本身没有任何功能，它仅是指向实现。接口可以跨组件复用和升级，可以增加新的方法。实际上，组件对外表现出来就是一个黑盒，对它的任何操作只能通过公共方法和属性组成的接口进行。理想的情况下，组件接口的规格说明，告诉组件的使用者组件可以做什么，而不必关心组件是怎样做的。一旦组件确定好以后，就要创建这个组件与其他组件之间的公共接口。在基于组件的软件系统中，对象是按照规范设计的模块，这些定义好的软件模块在系统中共存，并相互作用。组件系统的关键在于各模块的组合。每一个组件都有一个接口定义其属性及约束条件，因此各模块间接口的测试是整个软件系统性能的保证。

虽然关于组件的定义目前尚未有一个统一的标准，但对于组件本身所具有的核心属性的认识是基本一致的，那就是可重用性和可替代性。可重用性是指组件应该被设计成具有通用的功能，并且能在多个程序和工作环境中使用。可替代性是指一个具有等效功

能的组件可以替代现有的组件。随着组件技术的进一步发展，目前的组件技术又有了许多新的特点，如语言无关性、平台异质性及接口与实现分离等。一般而言，组件可以简单地是一个类或一个模块，也可以复杂如 EJB 或 COM 对象。组件比面向对象技术中的对象，具有更高的使用度，更灵活的生产方式，更容易理解和分发。

组件分为两类：

① 开发人员根据特定的组件标准专门开发，或原有的程序代码封装而成的内部组件；

② 由独立的第三方机构开发的商业组件。

7.1.3 组件的特点

我们知道，软件复用被认为是一条现实可行的途径，而组件就是软件复用用到的核心技术。通过对软件复用的分析、比较，发现软件组件有以下特点：

① 组件是一个包容的、可替换的软件单元，它封装了设计策略，并能与其他组件组合成更大的组件。

② 组件实现了对所提供服务的黑盒封装，其具体实现对外是透明的。

③ 组件提供并实现了一个或多个接口。组件必须通过文档化的接口进行访问，接口是组件与用户和其他组件之间发生交互的连接渠道，第三方只能通过组件的规格说明理解和重用组件。

④ 组件可以独立地被调用，并易于被第三方所组合。

⑤ 组件位置的透明性。组件以及使用它的程序能够在不同的进程或不同的机器上运行。组件对远程组件的处理方式应和对本地组件的处理方式是一样的。

⑥ 与语言无关性的组件，必须以二进制的形式发布，这样就将其实现所用的编程语言封装起来，做到与语言、开发环境无关。

⑦ 很好的可扩充性。由于每个组件都是自主的，只能通过接口和外界通信，当一个组件需要提供新的服务时，可通过增加新的接口来完成，不会影响到使用原接口的客户，而新的用户可以重新选择新的接口来获得服务。

⑧ 实现不同厂商的软件间的真正的互操作。组件可以来自不同的组件开发商，独立地被生产、获得和配置，不同的组件可以方便地搭建应用程序。

正因为组件的特点，给软件开发带来了好处，因此基于组件的开发和测试得到了广泛的重视。

7.1.4 组件的三个流派

在组件模型方面，已形成三个主要流派：对象管理联盟(OMG)的 CORBA、微软公司的 COM/DCOM、Sun 公司的 EJB。以下分别从 CORBA、COM/DCOM 和 EJB 三个方面来介绍当前软件组件技术的一些基本概念和内容。

为了协调和制定分布式异构环境下应用软件开发的统一标准，1989 年成立了一个国际组织——对象管理联盟(OMG)。加盟此组织的单位越来越多，现已有 750 多个单位，其中包括软件的开发供应商、软件用户和软件技术的研究院所等。经过多年的努力，已制定了一系列的标准规约，称为 CORBA(公共对象请求中介结构)。

CORBA 的核心是对象请求中介(ORB)，是分布式对象借以相互操作的中介通道。CORBA 核心 ORB 的作用是将客户对象(Client)的请求发送给目标对象(在 CORBA 中称为对象实现 object implementation)，并将相应的回应返回至发出请求的客户对象。ORB 的关键特征是客户与目标对象之间通信的透明性。CORBA 模型架构成熟，但它太复杂，未能被广大软件人员所接受，所以目前企业应用系统的核心平台，基本上都是选用 J2EE 和.NET。几大基础架构平台的代表是：基于 COM/COM+的.NET 平台、基于 EJB 的 Websphere 平台和 Weblgoci 平台。基础架构平台已超越中间件，成为实现企业应用软件开发、部署、运行、管理、集成和安全的一体化开放平台，可满足各种应用软件所要求的可靠性、可伸缩性和安全性的需要。

COM 是个开放的组件标准，有很强的扩充和扩展能力。COM 规定了对象模型和编程要求，使 COM 对象可以与其他对象相互操作。这些对象可以用不同的语言实现，其结构也可以不同。COM 规范包括 COM 核心、结构化存储、统一数据传输、智能命名和系统级的实现(COM 库)。COM 核心规定了组件对象与客户通过二进制接口标准进行交互的原则，结构化存储定义了复合文档的存储格式以及创建文档的接口，统一数据传输约定了组件之间数据交换的标准接口，智能命名给予对象一个系统可识别的唯一标识。统一数据传输建立在结构化存储的基础上，包括两方面的内容：首先是数据格式的统一，其次是传输协议的建立。DCOM 是微软与其他业界厂商合作提出的一种分布组件对象模型，它是 COM 在分布计算方面的自然延续，为分布在网络不同节点的两个 COM 组件提供了互操作的基础结构。DCOM 增强了 COM 的分布处理性能，支持多种通信协议，也加强了组件通信的安全保障。

1998 年 Sun 公司联合 IBM、Oracle、BEA 等大型企业应用系统开发商共同制订了一个基于 Java 组件技术的企业应用系统开发规范，该规范定义了一个多层企业信息系统的标准平台，这一规范和其定义的平台就构成了 J2EE。EJB(Enterprise JavaBeans)便是建立在 J2EE 平台基础上的企业级组件体系结构模型，是 J2EE 平台的核心，J2EE 的一个主要目的就是简化企业应用系统的开发，使程序员将主要精力放在商业逻辑的开发上。EJB 正是基于这种思想的服务器端技术，它本身也是一种规范，该规范定义了一个可重用的组件框架来实现分布式的、面向对象的商业逻辑。EJB 的核心思想是将商业逻辑与底层的系统逻辑分开，使开发者只需要关心商业逻辑，而由 EJB 窗口实现目录服务、事务处理、持久性、安全性等底层系统逻辑。

7.1.5 组件的形态

目前对于软件组件存在着不同的理解，这主要源于软件形态的多样性。众所周知，软件由程序和文档两部分组成。其中程序是以计算机语言表达的软件系统，文档是以人类

语言表达的软件系统。软件形态主要是指程序代码的形态，但不论哪一种形态的代码，都必须与文档相结合，才能构成完整的软件系统。程序既可以以方便人类阅读的源代码的形态(原码态)存在，也可以以计算机能直接处理的目标码形态存在(这里不涉及解释型语言，解释型语言不存在显式的目标码)。源代码经过编译即可得到目标码。目标码既可以以文件的形态进行存储(存储态)，也可以以进程的形态存在(运行态)。可以将源代码、存储态代码统称为静态代码。当需要软件提供具体功能时，则必须将其加载到一定的运行环境中，此时它就以运行态存在。存储态软件与运行态软件之间的关系类似于种子与树木之间的关系。

组件作为软件的有机构成部分，自然也存在多种形态。例如源代码形态的组件、目标码形态的组件，后者又可以分为存储态目标码组件与运行态目标码组件等。其中源代码形态的组件根据代码的结构组织方式和依赖方法又可分为以下几种形态：

① 类(class)：以类为单位进行封装而得到的组件，这是最基本的组件单元。

② 类树(class tree)：以一个抽象类为根，若干继承该抽象类的具体子类(也可能有抽象子类)为节点。将这样的类树封装为组件，抽象根类提供了该类树的对外接口，对具体子类的操作(删除或增加)以及子类对象的创建均由抽象类控制，实现了具体子类的隐蔽。

类树作为组件的优点在于：

- 类树比类具有更多的独立性，对外界依赖少，更易于复用；
- 类树封装使得用户只需要关心所需功能的抽象和规约而忽略具体子类细节；
- 类树封装后其结构变化和具体子类的增删对客户不会产生影响。

③ 框架(framework)：一个框架由一组协作组件组成。表明整个设计、组件间依赖及成员组件的功能分布，这些成员组件通常是子框架、类树或类，大多以抽象的形式出现，实现细节放在具体子类中，构成了一个抽象设计，不同的具体子类可产生对设计的不同实现。框架作为组件使得用户可以复用设计，用户通过具体子类的嵌入而在框架中加入特殊功能。

④ 设计模式(design parter)：是对经验的显式表示，每个设计模式描述了一个反复出现的问题以及该问题解法的核心内容，它命名、抽象并标识了一个通用设计结构的关键部分，使得它可以用来创建一个可复用的面向对象的设计。设计模式作为可复用组件体现了较高层次的复用设计思想。

⑤ 构架：应用系统体系结构的显式表示。构架具有领域相关性，组件根据构架进行复合而生成可运行的系统。构架是一类特殊的组件，可视为框架，用于描述一个应用系统时的极限状态。

上述各类形态组件间具有密切关系。通常一个构架由若干框架所构成，框架又可包含子框架、类树、抽象类和具体类，类树由抽象类和具体子类构成，类是最基本的构件单元。在这个意义上，组件的形态体现了组件粒度上的差异。一个框架通常含有多个设计模式的采用，每个设计模式都有若干个框架作为它在不同领域的具体实现。

7.2 组件测试

7.2.1 基于组件软件开发方法与软件测试

随着 Internet 的广泛应用和商业组件市场的快速发展，从面向对象的编程技术发展而来的基于组件的软件开发技术越来越多地应用于软件系统的开发，以提高软件质量和开发速度，从而使软件开发走上工程化和产业化的道路。其软件开发至少包括两个过程：一是复用组件开发过程，即开发可复用的软件组件；二是应用系统开发过程，即使用组件组装新系统的过程。这两个过程可以在时间与空间上相分离，因此这两个过程可以一并进行。这同传统的软件开发过程不同。基于组件的软件开发技术的最大特点是，用于构成软件系统的基础是由第三方提供的无源码的、基于不同平台和不同技术开发的组件，因此，面向组件的软件测试技术与面向对象的软件测试技术的要求有所不同。

面向组件的软件测试技术主要从以下两个方面分类：对组件开发者而言，是通过对组件的测试提供组件的质量证明，增加组件使用者使用该组件的信心；对组件使用者而言，是保证各组件间能正常有效地配合工作的集成测试，是否能正确完成系统要求的系统测试。同传统测试一样，在基于组件的软件开发过程中，测试也存在于每次迭代子过程中，目的是保证该迭代子过程中系统实现的功能与用户的业务需求模型一致。

传统上，在未与其他系统元素结合之前，大型系统的开发包含三个测试阶段。第一个阶段是单元测试，它关系到一个相对小的可执行程序。例如，在面向对象系统中一个单元可以包含一个类或一组逻辑上相关的类。第二个阶段通常指集成测试，这一阶段中为评价几个单元（已测试过）集成的系统，或子系统间的接口及交互而测试。第三个阶段是确认测试，集中于表现整体系统的功能或质量品质。这三个阶段按照已定义好的过程执行，与系统开发同步。图 7-1 展现了传统测试过程与基于组件软件测试的对比。

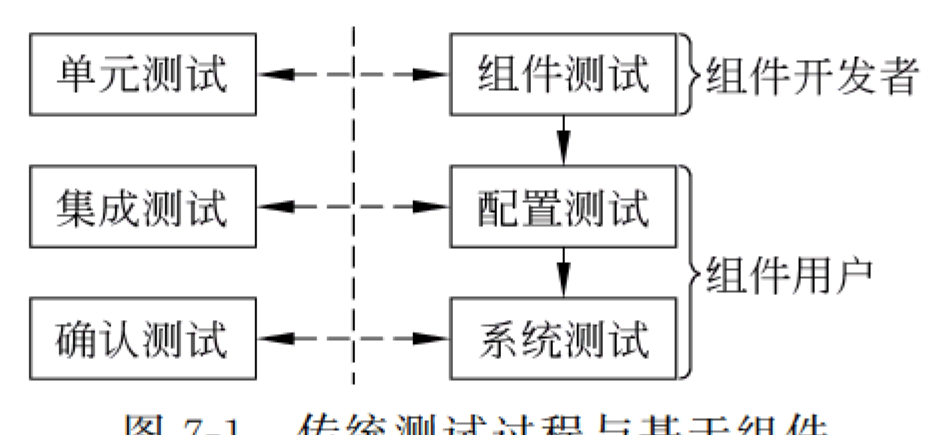


图 7-1　传统测试过程与基于组件软件测试过程的对比

在基于组件的开发中，三个传统的测试阶段应重新考虑及扩充。在这里组件便成为最小的测试单元。组件的开发者执行组件测试及构建合适的组件功能，同时需要在早期发现可能的错误。开发者建造的测试依赖于一个完整的文档说明、组件的知识、源代码的可行性及各种覆盖。但是，这样的测试不能将组件行为的功能定位到系统的详细描述中。

① 组件测试：通常由组件的开发者执行这一阶段的测试。它的目标是确认组件功能的可用性，并且更正早期的错误。使用 JUnit 这个工具可用来对组件内部进行测试。这个过程相当于传统测试过程中的单元测试。

② 配置测试：确认组成最终系统的组件的可行性。待确认的组件放到了实际运行环境中，并且与已有的组件共同组成系统的逻辑结构。

③ 系统测试：和传统测试中系统测试类似，在所有组件已集成好，且整个系统开始运行时，通常由用户执行。

在基于组件的测试过程中，对应于集成测试阶段用“配置测试”来表示。配置测试的目的是组成最终系统的组件执行的确认。在此阶段真正被确认的组件被放置在特殊的应用环境中，同时与现有组件集成来完成系统的逻辑结构。对于配置测试可以采用渐增的策略，允许将组件累积地集成到更大的子系统中。

集成测试的目的是尽可能地发现存在于协作关系中的软件故障。这些软件故障一般都发生在软件各部分行为的交互上，特别是对基于组件的软件系统问题更加突出。一般面向组件的基于 UML 的集成测试技术，主要是根据 UML 的动态图，再辅以从用例图或系统级的类图中提取必要的结构信息。

采用的测试技术大致分为两类：基于状态的测试技术和基于事件信息流的测试技术。这里事件信息流测试技术，类似于结构测试技术中的程序控制流或数据流技术，不同的是这里的事件信息流信息是从 UML 图（如序列图、协作图等）中获得，而不是从程序源代码中获得。

7.2.2 组件测试特点

从前面可以看到，基于组件的软件开发，不同于传统的软件开发和面向对象的软件开发，这将会给软件测试带来很大的影响。在传统的软件开发过程中，系统的构造、编码、测试是一个整体，通常开发团队进行了单元测试、集成测试、系统测试等全部测试步骤，测试过的软件系统可直接使用。而基于组件的开发包含两个过程：组件开发和组件集成。测试基于组件的系统主要分为两大方面的问题，即测试组件和测试系统。

围绕这两方面，测试应该从组件开发方和组件使用方两个不同角度考虑，这两方分别进行上述两方面的测试，测试的方式、策略和目的均有所不同。组件开发方负责通用组件的开发，组件即是软件产品。因此软件测试过程方面只需要进行单元（组件）测试。在基于组件的软件系统中，组件是基本构成单元，组件作为一个封装了数据和服务的模块，相对来说也是软件测试的最小单元。测试组件是确保基于组件的软件系统质量的基础，目的是为了提高用户对组件质量的信任程度，同时帮助用户减少测试费用。

对于组件的开发者（提供者）来说，在发布组件或将组件提交给使用者之前，必须把组件看作一个上下文独立的单元进行有效的测试，这相当于传统软件测试中的单元测试。由于组件开发者能够使用组件的源代码，所以可以使用结构测试技术和故障测试技术，另外功能测试技术和基于状态的测试技术也常常被组件开发者用于单个组件的测试。同时，由于生产组件的目的是重用，组件有可能被多种不同的应用环境所使用，因此需要在尽可能全面的场景中，单独地测试组件，确保组件正常工作。

组件使用方负责选择可重用组件、开发特定组件，再由组件组装成软件系统，他们从组件开发方得到经过测试的组件，但是无法保证这些组件捆绑在一起后能够正常交互，而且组成的整个系统能够正常使用。因为基于组件的软件系统故障一般都发生在各组件间的行为交互上，所以从组件使用者的角度来看，测试目的就是要验证，所选择的组件能够

和谐地工作。

组件使用方的测试又可分为两个步骤：组件集成测试和系统测试。

组件集成测试又包含结构集成测试和行为集成测试。结构集成测试主要测试组件的接口中有无错误；行为集成测试主要测试组件间交互是否被错误地实现，也称组件交互测试。组件之间的交互，可以通过直接的方式，也可以通过间接的方式。直接交互是通过接口实现组件间的交互，而间接交互就是通过消息传递的方式来实现交互。组件交互测试是对一组具有相互依赖关系的组件进行的测试。所谓依赖关系是指组件之间存在一定的调用、触发、激活等关系。构成系统的所有组件都能够根据这种关系来进行分组，每组组件应该能够独立完成一定的服务，因此需要对每组组件进行测试。

组件系统测试是测试由所有组件和主程序构成的整个系统，以验证软件系统的正确性和性能指标是否满足需求规格说明所指定的要求。这一步主要是在具体环境中进行功能测试、性能测试、强度测试等。基于组件的系统测试和传统的系统测试之间区别不大，可以借鉴传统的系统测试方法。

由以上分析可知传统的面向对象的测试理论和测试技术，可以用于测试单个组件，也可用于对基于组件的软件系统进行系统测试。然而，传统的集成测试技术却不太适合面向组件使用者的组件集成测试。这是因为：

① 组件的版权限制。组件的来源可能是 COTS 组件或第三方组件(共享软件、自由软件等)，因此一些组件的实现细节是不可见的，组件使用者几乎不可能获得组件的源代码，因此，基于软件代码的技术如结构测试技术和故障测试技术在这里无法直接使用。

② 异构问题。即使源代码是可见的，例如某些开放源代码的组件或团队内部开发的组件，也有可能是不同开发平台或使用不同程序语言开发的。使用的语言越多，测试的复杂性越大，有可能结构测试技术和故障测试技术也无法直接使用。

③ 由于组件常常不是根据某个具体最终用户的需求开发的，它提供的功能常常会比组件使用者所要求的更多。在这种情况下，用传统的测试覆盖率来衡量集成测试结果是没有意义的。因为有可能组件提供的某些功能在该系统中根本不会被使用。例如，在结构测试中的基于程序控制流图的技术评估标准是：程序中所有控制流的测试覆盖率要求达到某个特定的百分比。但是在一个基于组件的系统中，评估这样的测试覆盖率，应该要排除组件在目标系统中不使用的功能部分，否则即使一个设计很完善的测试集，也可能只获得较低的测试覆盖率。

可以看到，组件集成测试因为其重要性和特殊性，应成为组件使用方首要关注的问题。假设组件内部已经通过了测试，这样基于组件的软件系统的可靠性，关键就在于这些组件间的交互，下面将着重讨论从组件使用者角度出发，如何对组件进行集成测试。

7.2.3 UML 在组件测试中的引入

从前面的分析可看出，组件测试技术需从开发方和组件使用方不同角度进行讨论。组件开发方的测试工作针对两个主要方面：一是采用传统方法对单个组件进行测试，使得用户能够得到正确、可靠的组件，减轻用户测试负担；二是组件开发方应对用户提供必

要的测试支持，例如可以采用一定的分析技术收集组件的概要信息，并随组件发布，这样用户虽然得不到源代码，但这些组件概要信息可以使他们进一步分析、测试组件系统变得容易一些。很多文献描述了程序分片、控制依赖分析等几个方法，为组件用户提供测试支持的方法。

而组件测试技术的重点在用户方，用户对组件进行充分的集成测试更为重要，因为单个组件的正确并不能保证它们集成在一起就能正确运行。

组件的特殊性质给测试工作带来很大挑战。Weyuker 说，随着组件重用和使用第三方组件越来越普遍，我们需要一种能够广泛应用并且不需要源代码的测试技术，因为通常只有对象代码是可得的。

目前，对组件系统的集成测试，主要采用的是基于黑盒方法的功能测试。即从组件或系统的外部要求和特性出发，从各种不同的角度对其进行全方位的测试，包括基本功能、接口、非法输入、相互调用、时间和空间性能、兼容性等。

组件的规格说明是组件开发的基础，同时也是组件测试的基础。首先来了解一下组件的规格说明。理想情况下，组件接口的规格说明告诉组件的使用者，组件可以做什么，而不必关心组件是怎么做的。此外，它还应该给出组件应满足的约束条件，提供一种机制使用户可以明确使用的组件是否可以在一定的上下文中被使用，以及怎么去使用。而组件系统的规格说明描述了各个组件之间如何交互来实现特定的系统功能。同时还有对组件系统结构方面的约束描述，但这不是组件集成测试的重要基础，主要关注基于描述组件行为和交互方面约束的规格说明的测试。

目前，采用纯手工方式选择测试用例，使得测试的成本居高不下，因此测试用例的自动生成方法研究就具有重要意义。同时，为了保证测试工作的充分性和有效性，以最小的代价获得尽可能好的测试用例，提出测试覆盖准则也很重要。可以说对于组件测试，最主要的问题是测试用例的自动生成和测试覆盖率问题。

黑盒测试生成测试用例的依据是软件系统的规格说明，即编码前的分析设计模型/文档。而系统需求和分析阶段的规格说明，一般采取纯自然语言编写，形式化程度低，规范性差，内容涉及面广泛，难于给出一个系统化、自动化的测试方法，也无法提出有效的覆盖率指标，难于判定测试的充分性和完整性。

为此，寻找一种统一的、使用广泛而相对形式化程度又不过高的规格说明描述语言，作为测试用例生成的基础，是组件测试急需解决的问题。而软件工程领域最近几年当中最重要而且具有划时代意义的成果就是 UML 的出现，它的作用范围不仅支持面向对象的分析和设计，还支持从需求规格描述开始的软件开发的全过程。所以，把基于规格说明的测试用例生成建立在 UML 模型的基础上。UML 作为 OMG 的标准建模语言，已被业界广泛采用，并有大量成熟的可视化建模工具，用于从软件需求分析到设计实现部署的各阶段，从而在广泛性、形式化和自动化方面为解决以上问题提供了一个契机。

UML 作为建模语言事实上的标准，近年来被学术界和工业界广泛地用于软件系统建模。从测试角度看，这些模型是获取系统结构和行为信息的来源，因而是测试用例生成的理想基础。具体地讲，UML 模型在指导测试方面有如下优点：

① 通用性：UML 作为标准建模语言，具有广泛的适用性，无论软件采用何种语言编

写，都可以使用UML建模。目前UML已被软件开发界广泛采用，支持从软件需求分析到设计实现部署的各阶段，也很容易从组件开发方得到相应的UML模型图。同时有大量商业工具支持。

② 形式化：UML模型具有严格的定义，提供了对象结构和行为的表示方法。这种形式化特性使得测试信息的提取和自动化变得容易。

③ 描述能力：UML模型集众家之长，具有强大的描述能力。UML包括一系列视图和模型，它们从不同的层次和角度描述了软件系统的结构、行为以及软件的使用，可以满足现实世界不同层次的需要。

④ 管理能力：UML模型通过提供不同层次的视图和包机制等，具备了强大的管理能力，解决了模型维护和管理的问题。同时通过分层等方法在一定程度上解决了状态空间爆炸的问题。

⑤ 可重用性：UML模型支持在软件开发的各阶段，从不同的抽象层次对系统各方面的相关信息进行建模。这些模型不仅可以用于软件开发阶段，还可以用于指导测试，因此避免了专门为测试构造模型，实现了软件分析和设计阶段制品的重用，同时也将测试活动与开发过程集成起来。

⑥ 可迭代性：可以尽早开始测试活动(包括测试计划的制定、测试大纲与测试用例的设计等)，并随着设计活动的不断细化生成的测试制品。这样，软件开发与测试开发可以并行进行，并在整个测试过程中进行持续测试活动。众所周知，越早开始测试越好。

基于以上原因，UML不仅是软件开发的重要工具，同时也是指导测试的重要模型，在基于组件的软件系统的测试中可以发挥重大作用。

7.2.4 组件测试方法

目前UML提供9种不同的图，可分为两大类，即静态图和动态图。静态图用于描述系统的静态结构，包括用例图、类图、对象图、组件图和配制图；动态图是对系统动态行为的建模，包括序列图、协作图、状态图和活动图。使用不同的UML图可以从各角度描述软件系统的功能和内容。

由于软件测试的主要目的是尽可能地寻找软件系统执行过程中的缺陷或错误，所以大多数的软件测试技术是基于UML动态图为主，辅以静态图提供必要的基本信息。

UML的应用贯穿于整个系统开发的各阶段，在需求分析阶段，使用用例视图表示客户需求；在分析阶段，用逻辑视图和动态视图来描述；在设计阶段，把分析阶段的结果扩展成技术解决方案，具体产生构造阶段的详细规格说明；在构造阶段，根据详细规格说明生成程序代码；在测试阶段，根据不同测试阶段使用不同的UML图，以判断被测试的部分或软件系统是否与设计相符或有哪些偏差。对于面向组件的软件测试，也可以按照传统方法分为3个测试级别：单元测试、集成测试和系统测试。UML图可以分别用于不同的测试阶段。

面向组件的集成测试技术主要是根据UML的动态图，再辅以从用例图或系统级的

类图中提取的必要的结构信息，生成测试用例。采用的测试方法可大致分为两类：基于状态的测试技术和基于事件信息流的测试技术。

1. 基于状态的测试技术

基于状态的集成测试主要是从状态图中提取测试相关信息。其中一种方法是先从各UML状态图产生一个系统级的全局组合状态图，之后根据自底向上的集成策略对各部分集成，然后分别测试，以发现软件各部分（包括接口部分）合作是否会产生故障。这种方法的缺点是存在状态爆炸问题。

状态爆炸问题

状态图最早是作为一个图形化的表示方法而被提出的，Harel在此基础上增加了通信、并发和嵌套机制，使之更适合于描述复杂系统。最近研究较多的集中于UML形式化描述状态图。

UML状态图本质上是一个扩展的有穷自动机，它与传统自动机的区别在于，UML状态图支持并发、层次化、事件等特性。对于状态图的测试，通常的方法是由状态图生成相应的积自动机，然后采用有限状态机的测试序列生成策略。积状态机的状态是所有并发过程的状态组合，即并发过程的状态的笛卡儿集，如果M1、M2表示两个通信状态机，$|M|=|M1|\times|M2|$，所以积状态机的状态数量是所有并发过程的状态数量的乘积，就好比90 000＋90 000得到的结果没什么，但90 000×90 000得到的数量就很吓人了，起码差了几百万个数量级。这样的数量即使对于一个较小的系统也是难以接受的。这就是所谓状态爆炸问题。

一种解决状态空间爆炸问题的办法是对系统的各个部分分别采用状态机建模，即采用一组状态机为整个系统建模。

对于组件的动态行为可以使用状态图来描述。状态图记录了组件所有可能的状态以及状态转换所需要的条件。状态转换发生在从对象的接口中接收到事件的处理上。状态图是对系统的动态行为进行建模，它的层次结构和并发机制使得它对大型系统的建模更加容易。

对于状态图的测试，通常的方法是由状态图生成相应的状态监控测试程序，通过测试状态随输入条件不同而发生变化，检查整个过程是否符合状态图。几个状态图进行集成时，生成一个局部组合状态图，此时，每个状态图都可被视作可信任的单一状态，此时对状态图间的状态转化进行测试就可以完成对局部组合状态图，自底向上，最后一直组合成全局组合状态图。

2. 基于事件信息流的测试技术

基于事件信息流的测试技术类似于结构测试技术中的程序控制流或数据流技术，不同的是事件信息流信息是从UML图中获得，而不是从程序源代码中获得。从数据流、控制流生成测试用例的方法可以用在面向组件的测试中，从UML模型提取相应信息生成测试用例。下面介绍一个具体方法：

首先提取出所有的事件信息流的所有流向路径,采用类似于树的形式保存每个事件信息流的所有可能结果,这样就生成了测试用例。对 UML 模型中的所有事件信息流都构造了测试用例,就形成了事件信息流测试用例集。由于数据在程序中,实际上还是以结构化的形式进行传递和处理,这种处理方法,可以完全遍历测试事件信息的处理。通过比较应该发生的方法流向和实际执行时观察到的方法流向是否一致,可以检测确定该处处理是否正确,从整体上对 UML 图描述的系统行为的实现是否与设计一致,从而完成集成测试。

7.3 运行时分析技术

7.3.1 运行时分析定义

运行时分析是通过在调试器中运行目标程序,观察执行过程中的代码,以发现潜在问题。它与“静态分析”不同,静态分析是指通过观察源代码或高级架构分析系统行为,或是分析系统失败时的“崩溃分析”。

运行时分析是一种动态的检测技术,从代码流和数据流两方面入手:通过设置断点动态跟踪目标程序代码流,以检测有缺陷的函数调用及其参数;对数据流进行双向分析,通过构造特殊数据触发潜在错误并对结果进行分析。

其实程序员每天都在接触运行时分析,调试就是一种经典的运行时分析的例子:设置断点,一步步地运行系统,并且观察评估值和变量,通过这些方法,了解系统运行时所发生的情况。调试的过程中,程序员不会对所有可能发生的事情感兴趣,仅对真正发生的事情感兴趣。这正是将运行时分析从静态分析中分离出来的原因。

运行时分析需要借助调试器,在调试器中运行程序,便于观察程序运行状态、内存使用状况以及寄存器的值等,这些信息对于 Bug 的发现与分析都非常重要。它有利于发现实时环境中的异常,能精确显示错误发生时的上下文,有利于精确定位错误,但同静态分析类似,仍需要大量时间、精力和实验,要求分析人员有很好的编程语言功底。

调试是传统运行时分析的一项高度互动且有效的功能,但它受限于屏幕所能展现信息的数量。当关注的焦点集中于程序的一行或一个变量时,它很难识别出某些整体上的错误。因此调试虽然可以检测出很多问题,但一个通过了完整调试并获得正确结果的系统,仍有可能存在严重的质量问题。这就需要另外一种运行时分析:通过数据或事件来检验系统是否正确的运行。这种类型的测试是软件质量的基础。

运行时分析扩展了一种软件开发行为:关注质量。它通过更好地理解应用的内部工作过程实现了更高的软件质量。可靠准确地运行时,性能、内存使用和线程与代码覆盖分析数据,是唯一能决定应用程序不含严重错误并且可以高效执行的标准。

7.3.2 运行时分析分类

运行时分析主要分为四类，为了方便说明，这里引用 PurifyPlus 工具作为例子。

① 系统开发周期中的运行时分析。交互的开发周期和工程质量方案中，重要的自动化构建、测试过程中都会包含运行时分析。

② 开发与执行过程中的运行时分析。在交互开发过程中，Purify 能够在新代码加入工程前报告内存错误。开发人员能够使用 PureCoverage 识别所需新测试的区域。Quantify 能够在开发阶段早期发现测试人员不期望的代码问题和瓶颈问题。

③ 自动化测试过程中的运行时分析。当运行自动化构建测试时，PurifyPlus 能够监测质量并且收集信息，以确保工程运行于正轨。当测试运行于 Purify 且没有错误报告时，这时就可以知道，系统并不含有内存访问错误和内存泄露。当所运行的 Quantify 显示性能满足目标时，这时就可以知道，系统没有遇到瓶颈。当 PureCoverage 报告高级别的代码覆盖时，这时就可以知道，并没有引入新的代码块，因此不必增加自动测试。

可以增加利用自动化测试工具(如 IBM Rational Functional Tester)所获得的运行时分析数据的质量。自动化测试能够在一次迭代中得到更多的测试信息，并且能够评估所引入的新产品质量效果。如果软件质量介于两类连续的组件迭代之间，那么运行时分析数据很容易找出其中的特点或代码的改变。

④ 用以理解系统的运行时分析。有时运行时分析并不包括正确性或质量分析。例如，运行时分析可以帮助理解系统，特别是 Quantify 可利用调用图，显示系统的各个部分如何协调工作，或发现测试人员不期望的性能下降等问题。

7.3.3 关键运行时参数的测量

可见的错误检测仅是运行时分析的第一步，还需要准确理解运行时所发生的事情。因此，运行时分析应基于针对应用执行的准确的参数测量：运行时性能 、内存使用、代码覆盖。

1. 测量实例

(1) 测量实例 1：函数列表视图

函数列表视图是一种典型的运行时分析视图，它由特殊的运行时分析工具(如 Rational Quantify)创建。图 7-2 显示了表中全部重要的方法和/或应用对象。这样就允许开发人员通过分析代码，找出使用内存的最佳方法，同时找出运行最慢的函数、对象生存时间等。这个视图还提供了关于方法调用总数、所选方法的花费时间及其全部派生的准确信息。

(2) 测量实例 2：函数 Function 详细视图

运行时分析工具如 Rational Quantify 还能够扩展测量实例 1 中的信息，使它包括了

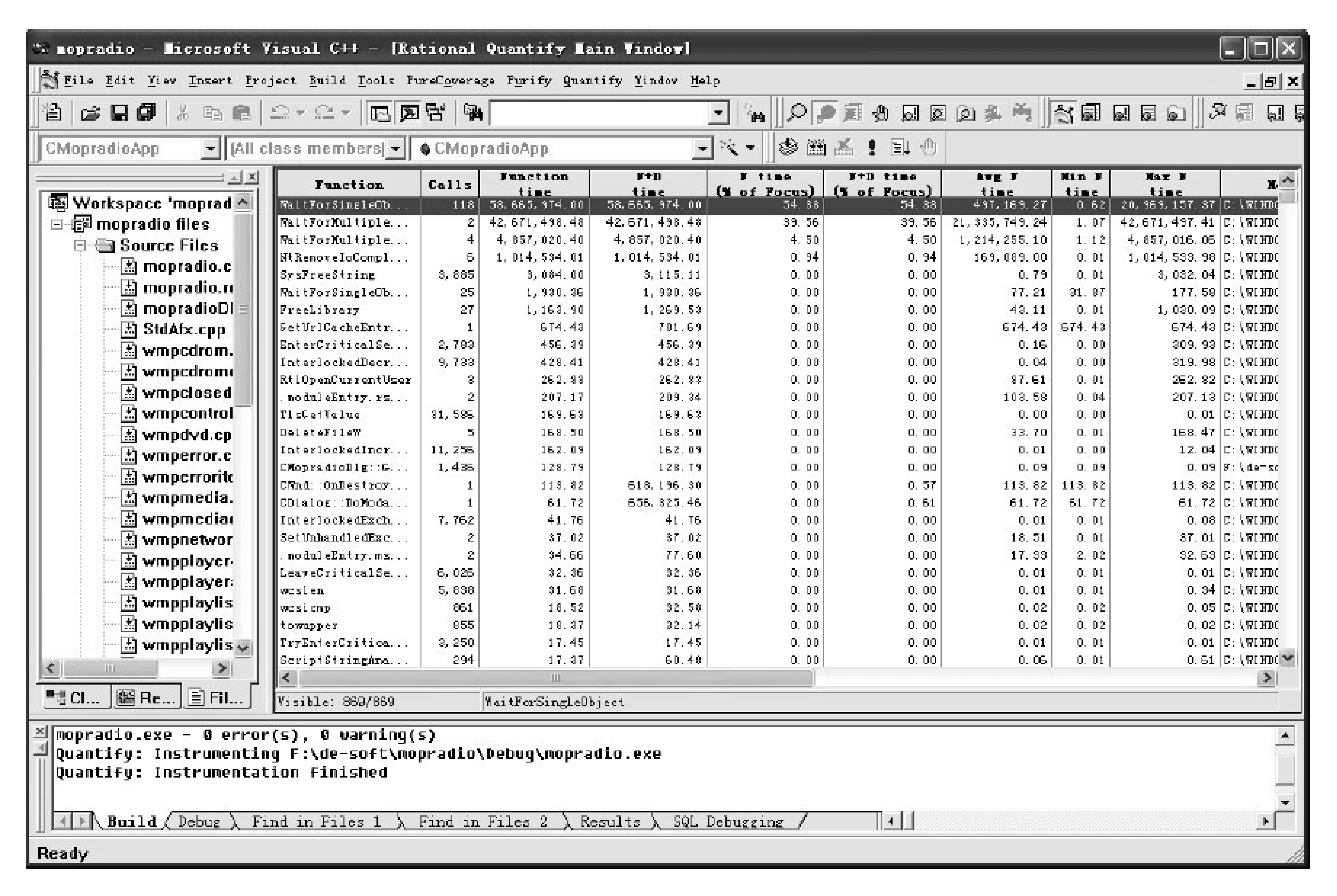


图 7-2　针对 Visual C++ Application 的 Rational Quantify 函数列表视图

调用方法和派生时间的测试数据的分布信息，如图 7-3 所示。这一视图强调了对于性能或内存热点（能够用以帮助检测准确的性能或内存瓶颈的原因的信息）有意义的调用方法和派生。

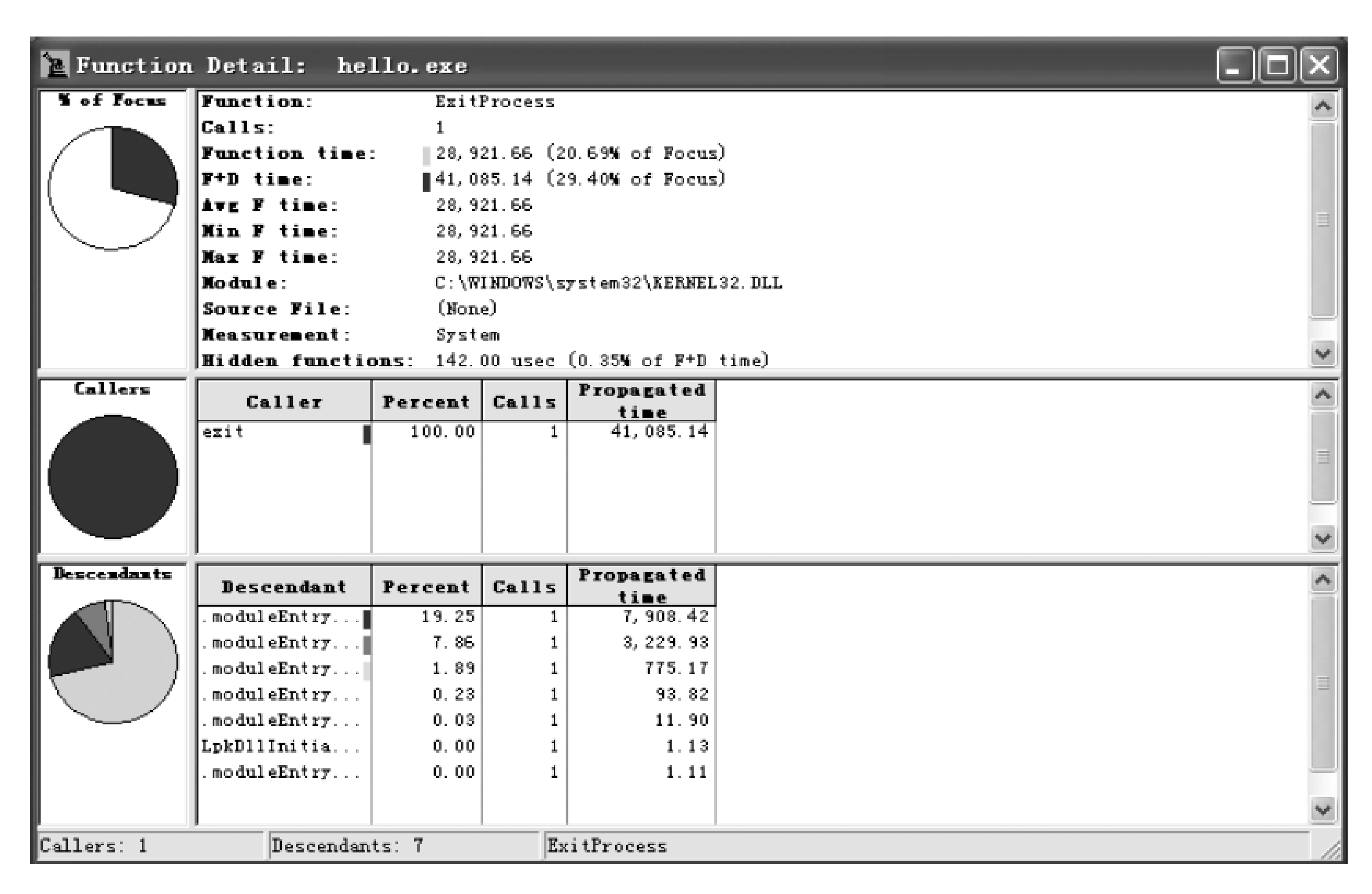


图 7-3　Visual Studio.NET 中的针对 Visual C#.NET 的 Rational Quantify 函数详细视图

（3）测量实例 3：方法覆盖模块视图

在评估可用测试方法价值时，对于测量测试过程所覆盖的代码比率或简单的标记，所有未测试的方法是十分有用的。这时可以应用诸如 Rational PureCoverage 的工具，用以

产生准确的未测试和废弃代码与已测试代码的信息，如图 7-4 所示。

图 7-4　Visual Studio. NET 中对于 Visual C#. NET 和 Visual Basic. NET 应用程序的混合方法代码覆盖的 Rational PureCoverage 显示

2. 错误检测：用户模式下的运行时内存崩溃

这是对于纯 C/C++ 应用的运行时分析的最大好处。运行时分析不但能通过显示性能、内存、线程和不同视图中的代码覆盖数据检测错误，还能够在用户模式下，准确地指出产生和/或引起错误的位置。运行时内存崩溃检测能够确保所有平台上的正确函数和高质量的纯 C/C++ 应用程序。对于运行时内存检测的 IBM Rational 工具有 Rational Purify 和 Rational PurifyPlus。下面举几个例子。

（1）错误检测实例 1：Rational Purify 内存错误和内存泄露报告

Rational Purify 能够指出开发人员创建的内存错误的准确代码行，甚至不需要源文件就可以提供这一信息。Rational Purify 在内存中检测错误，并使用调试信息追踪回溯找出相应的代码行，如图 7-5 所示。

在这个例子中，开发人员忘记构建数组变量时考虑终止符。这一错误将会导致应用构造的崩溃，而调试构造却很正常。本例仅是运行时分析减少 C/C++ 开发调试时间的一个例子。

（2）错误检测实例 2：量化标注源

Rational Quantify 具有一种独一无二的能力，它能够为用户方法代码的每一行测量记录的时间分布。量化标注源显示了每行代码的测量时间，和调用这行所花费的时间。这类信息能够有助于将性能瓶颈缩小到代码行级别，如图 7-6 所示。

（3）错误检测实例 3：Purify 对象和指针图

在 Java 和. NET 管理代码中，不可能产生诸如读写边界溢出和读写释放内存的运行

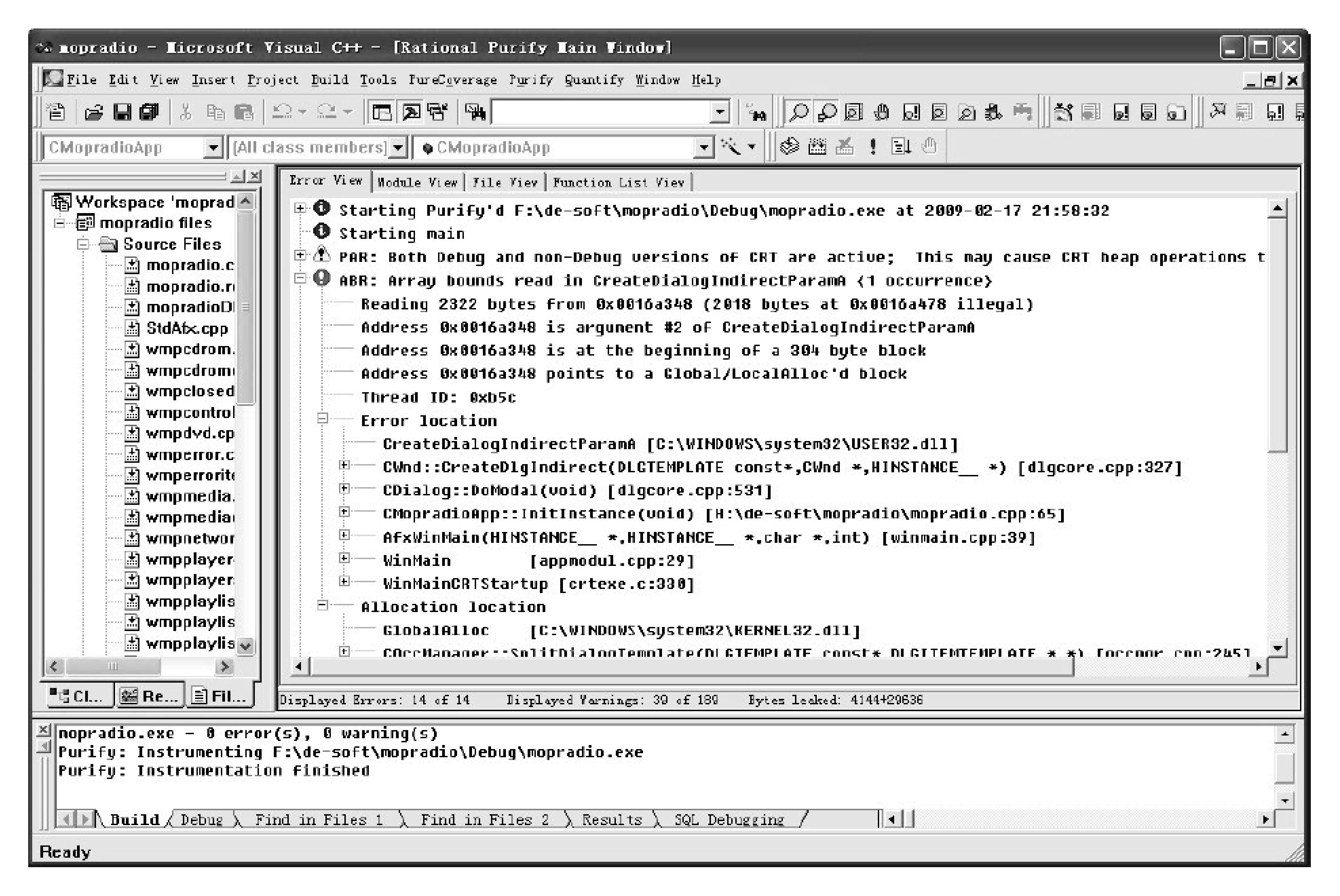


图 7-5　针对 Visual C++ 应用的 Rational Purify 内存错误和内存泄露报告

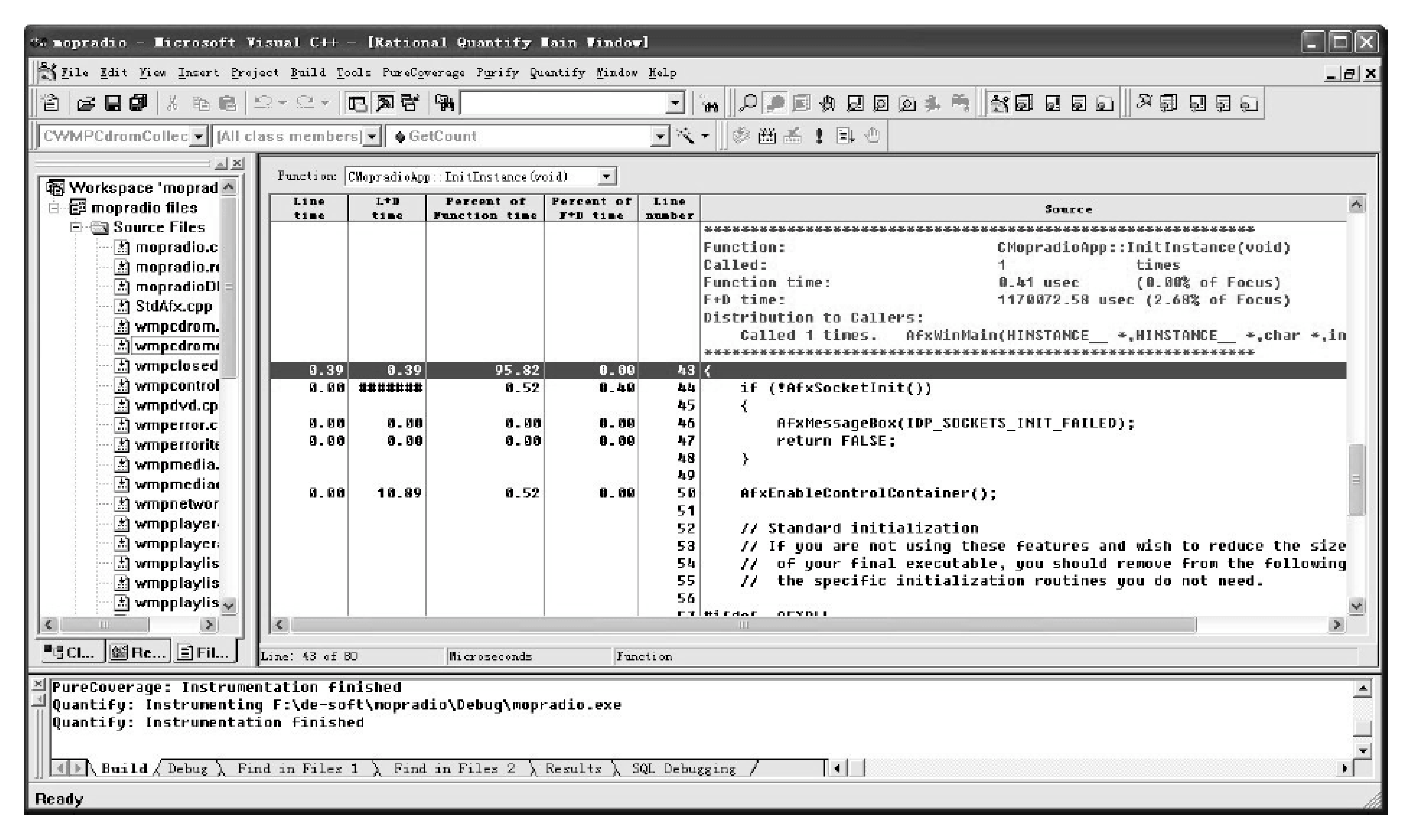


图 7-6　Visual Studio 6 中针对混合的 Visual Basic 6 和 Visual C++ 应用的 Rational Quantify 标注源

时内存错误，因为运行时子系统的自动内存管理机制禁止开发人员直接访问所分配的内存。但如果程序员忘记对象指针分配内存，这种自动化的内存管理机制就无能为力。只要代码中存在这种动态分配对象的指针，他们将存在于内存中，并不会被自动化的内存管理（垃圾回收技术）所清除。这种错误的影响与 C/C++ 中的泄露是一样的：内存对于其

他运行于主机操作系统中的进程变得不可访问。但是通过使用 Rational Purify 的运行时分析，测试人员能够准确定位发生问题的对象指针的代码行，如图 7-7 所示。

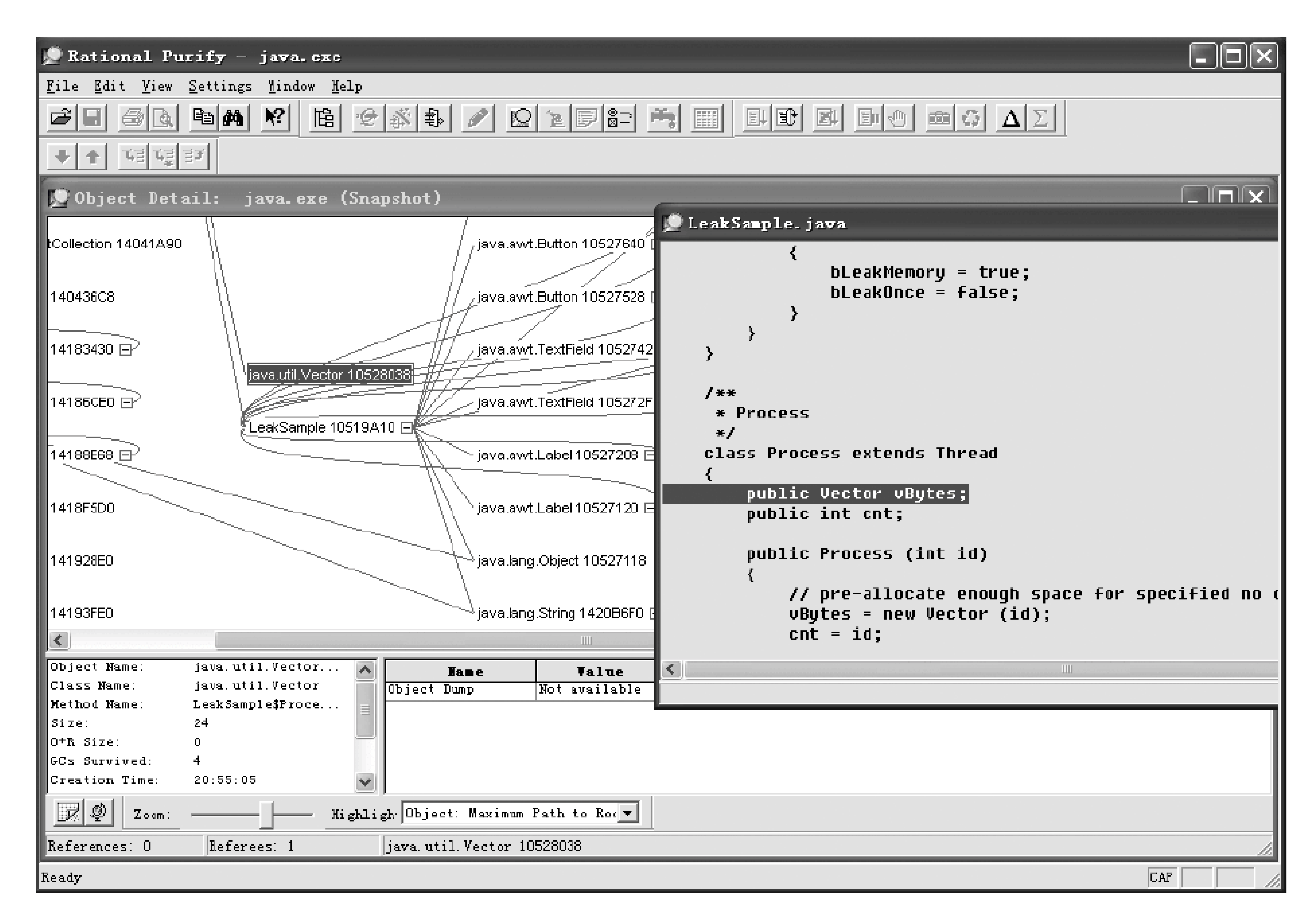


图 7-7　针对 Java 应用的 Rational Purify 对象和指针

7.3.4　运行时分析的文档

另一种运行时分析方式是对运行时分析的文档化操作，这样可以帮助评估工程的全部质量，并评估新引入特征对应用性能、可靠性、完整性所带来的影响。这种实际运行时分析的方式包含了开发和分析数据下，对每次组件或应用迭代的收集运行时数据。这些信息有助于确定工程质量附加新特征与错误定位的效果。

当使用了运行时分析工具，可以很容易地检测到源代码数据库的变化，只要这些变化与错误的构造和/或自动化测试的失败有关。测试人员能够发现在成功测试与失败测试间的源代码库的变化，还能够识别那些改变代码的开发人员和引入错误的准确时间和日期。

高级的运行时分析工具(如 Rational PurifyPlus)提供了分析多重测试运行的功能，例如，允许用户混合各种不同测试的代码覆盖数据或为连续测试迭代创建单独的数据集合，如图 7-8 所示。

在图 7-8 中，Rational Quantify 对比了两类数据集，并强调了性能改进的调用链(绿线，图的下半部分)和性能下降的调用链(红线，图的上半部分)。最后的数据可以从 Call Graph 视图和更详细的 Function List 视图中获得。

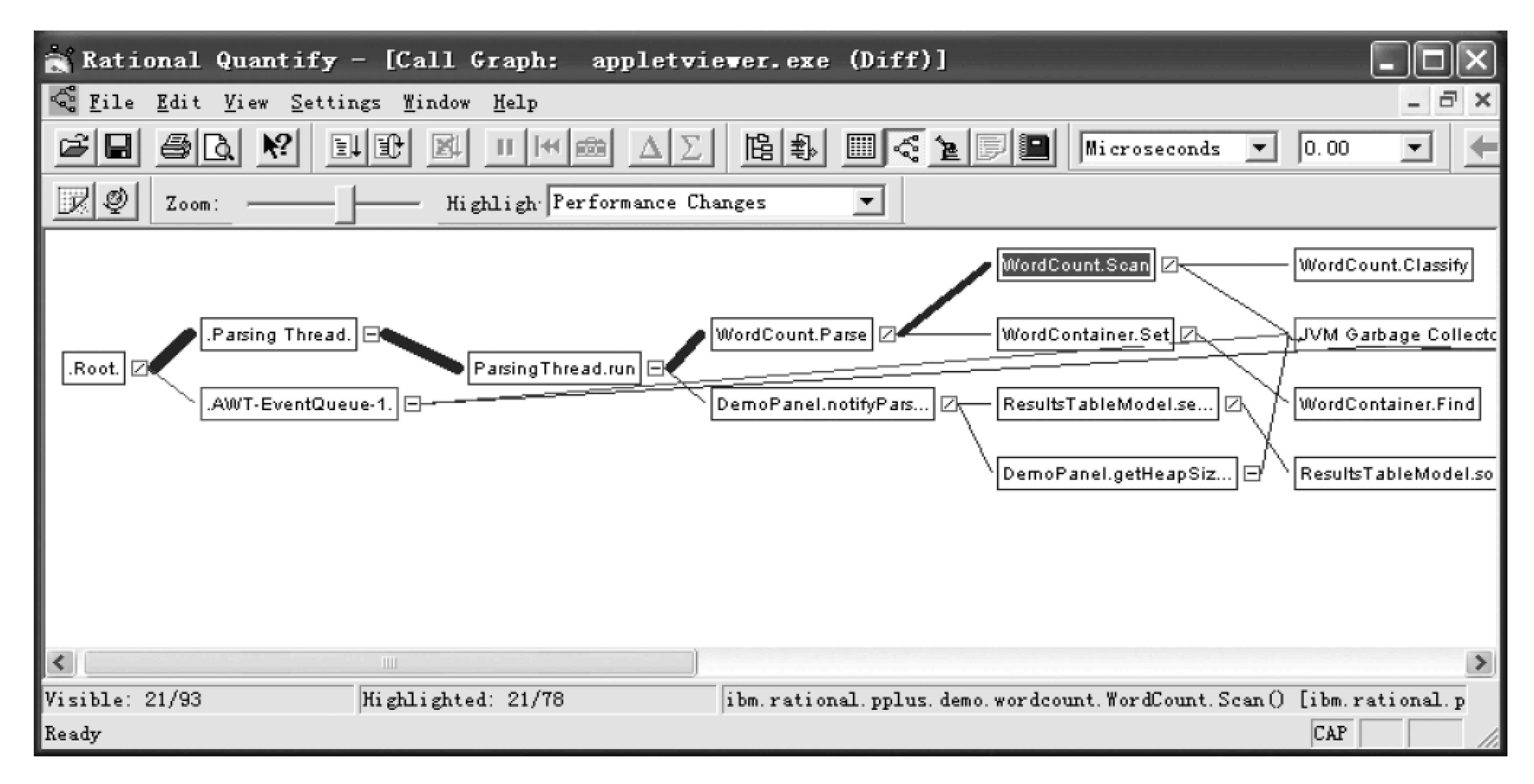


图 7-8　Rational Quantify 比对运行报告

即使没有创建自动化的测试环境，测试人员仍然可以通过利用存储为 ASCII 文件的运行时分析数据创建自动化的数据分析。图 7-9 显示了一个导入到 Microsoft Excel 中的性能概况的例子。

Microsoft Excel - 图7-9.xls

	Function Time	Calls	Function	F+D time	F time (% of focus)	F+D time (% of focus)	Avg F time	Min F time	Max F time	Module	Source File	Measurement	Hidden Functions
2	167949099.6	3970	WaitForSingleObject	167949099.6	29.9	29.9	42304.56	0.44	11924384.89	C:\WINDOWS\s	(None)	Sys Block	No
3	126386287.5	11635	WaitForMultipleObjects	126386287.5	22.5	22.5	10862.59	0.48	39636358.13	C:\WINDOWS\s	(None)	Sys Block	No
4	76765029.76	13	WaitForMultipleObjectsEx	76765029.76	13.67	13.67	5905002.29	0.97	51698811.84	C:\WINDOWS\s	(None)	Sys Block	No
5	13522448.21	483	WaitForSingleObjectEx	13522448.21	2.41	2.41	27996.79	0.58	13515742.47	C:\WINDOWS\s	(None)	Sys Block	No
6	5960321.4	3825	GetQueuedCompletionStatus	5960321.4	1.06	1.06	1558.25	0.01	15242.33	C:\WINDOWS\s	(None)	System	No
7	4352404.89	70240	EnterCriticalSection	4352404.89	0.77	0.77	61.96	0	2499374.52	C:\WINDOWS\s	(None)	Sys Block	No
8	3137253.57	3	NtRemoveIoCompletion	3137253.57	0.56	0.56	1045751.19	0.01	3137251.5	C:\WINDOWS\s	(None)	System	No
9	128648.1	73204	LeaveCriticalSection	128648.1	0.02	0.02	1.76	0.01	13155.98	C:\WINDOWS\s	(None)	System	No
10	127668	1280	SetTimer	127668	0.02	0.02	99.74	3.87	4404.88	C:\WINDOWS\s	(None)	System	No
11	108409.63	1	CWnd::OnDestroy	222780.24	0.02	0.04	108409.63	108409.63	108409.63	C:\WINDOWS\s	(None)	Timed	Yes
12	67709.75	10768	GetTextExtentExPointWPri	67709.75	0.01	0.01	6.28	0.01	10770.89	C:\WINDOWS\s	(None)	System	No
13	67366.77	1121	SetThreadPriority	67366.77	0.01	0.01	60.1	0.45	15547.37	C:\WINDOWS\s	(None)	System	No
14	66676.17	2	.moduleEntry.rsaenh	66485.99	0.01	0.01	33338.08	3.92	66672.25	C:\WINDOWS\s	(None)	Timed	Yes
15	64929.79	68	initterm	66064.61	0.01	0.01	954.85	0.22	37814.27	C:\WINDOWS\s	(None)	Timed	Yes
16	57089.98	3075	DeviceIoControl	57089.98	0.01	0.01	18.57	3.13	2416.29	C:\WINDOWS\s	(None)	System	No
17	48018.69	119	CoCreateInstance	242869.48	0.01	0.04	403.52	3.84	15897.86	C:\WINDOWS\s	(None)	Timed	Yes
18	40743.19	847	SetEvent	40743.19	0.01	0.01	48.1	0.5	9361.83	C:\WINDOWS\s	(None)	System	No
19	39752.05	2	.moduleEntry.NETSHELL	45180.99	0.01	0.01	19876.03	8802.75	30949.3	C:\WINDOWS\s	(None)	Timed	Yes
20	37350.4	2	.moduleEntry.wmp	83490.82	0.01	0.01	18675.2	229.58	37120.84	C:\WINDOWS\s	(None)	Timed	Yes
21	36749.48	3	InternetGetConnectedStateExA	56356.73	0.01	0.01	12249.83	5.59	36734.36	C:\WINDOWS\s	(None)	System	Yes
22	36190.39	8	CreateActCtxW	36190.39	0.01	0.01	4523.8	371.38	6380.23	C:\WINDOWS\s	(None)	System	No
23	30447.95	27	COleDispatchDriver::InvokeHelper	80183.4	0.01	0.01	1127.7	5.68	13795.55	C:\WINDOWS\s	(None)	Timed	Yes
24	29359.16	79	.moduleEntry.ESENT	33010.01	0.01	0.01	371.63	0.29	15406.92	C:\WINDOWS\s	(None)	Timed	Yes
25	28610.53	1	WMCreateReaderPriv	35602.67	0.01	0.01	28610.53	28610.53	28610.53	C:\WINDOWS\s	(None)	Timed	Yes
26	26092.62	10979	ReleaseMutex	26092.62	0	0	2.38	0.48	9520.63	C:\WINDOWS\s	(None)	System	No
27	26000.69	1630	ScriptStringAnalyse	119547.42	0	0.02	15.95	1.27	12572.01	C:\WINDOWS\s	(None)	Timed	No
28	22793.79	1	RpcMgmtInqServerPrincNameW	22961.64	0	0	22793.79	22793.79	22793.79	C:\WINDOWS\s	(None)	System	Yes
29	20618	2	.moduleEntry.quartz	20891.05	0	0	10309	9.52	20606.48	C:\WINDOWS\s	(None)	Timed	Yes
30	17641.74	304	CoGetObjectContext	21857.23	0	0	58.03	0.37	17346.37	C:\WINDOWS\s	(None)	Timed	Yes
31	17051.48	11188	InterlockedExchange	17051.48	0	0	1.52	0.02	15806.11	C:\WINDOWS\s	(None)	System	No
32	16783.1	2	.moduleEntry.DBGHELP	17804.89	0	0	8391.55	613.37	16169.72	C:\WINDOWS\s	(None)	Timed	No
33	16145.92	1	CDialog::DoModal	2137983.75	0	0.38	16145.92	16145.92	16145.92	C:\WINDOWS\s	(None)	Timed	Yes
34	15816.69	98	FreeLibrary	81008.61	0	0.01	161.39	0.15	5378.68	C:\WINDOWS\s	(None)	System	No
35	15625.42	14	InternetGetCookieW	151687.31	0	0.03	1116.1	22.5	15280.85	C:\WINDOWS\s	(None)	System	Yes
36	15618.23	2	DDX_Control	15630.42	0	0	7809.11	3.59	15614.64	C:\WINDOWS\s	(None)	Timed	Yes
37	15585.98	32	iertutil.dll.#653	18402.98	0	0	486.44	0.08	15036.93	C:\WINDOWS\s	(None)	System	No

Sheet1 / Sheet2 / Sheet3

就绪

图 7-9　Rational Quantify 性能报告被导入到 Excel 文件中

这样可以编写简单的 Visual Basic 程序以读取 Excel 文件来实现自动化的数据分析，或以 Perl 或 JavaScript 等脚本语言编写脚本来管理和分析来自各种测试的数据。

7.3.5 运行时分析例子

运行时分析已经在 Rational 系列工具和.NET 里实现，例如 Rational PurifyPlus 就可以进行运行时分析。

1. 代码覆盖

使用诸如 PureCoverage 的运行时工具，可以看到各种视图显示代码覆盖信息，其中之一就是 Annotated Source。这一特别的视图显示了所检查的应用的源文件，指出了执行测试事例之后的状态：hit、missed 和 dead。

2. 线程

运行时分析工具诸如 Rational Quantify 提供了线程可视化，它能够利用标记调试时检测每个线程的状态，检测多线程问题。如图 7-10 所示，它允许在调试时检测可视化线程的状态。

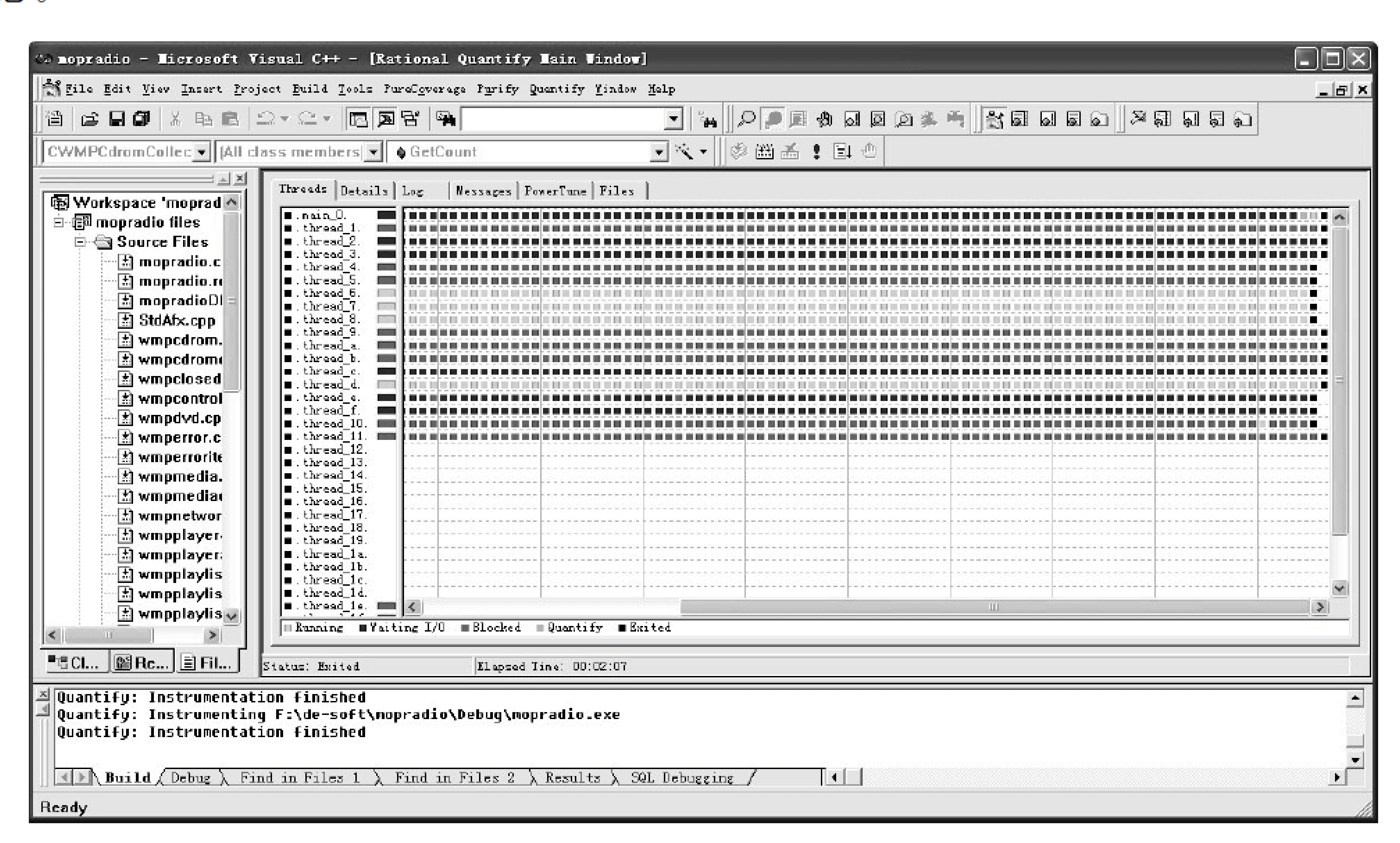


图 7-10　Visual Studio 6 中的 Rational Quantify 线程分析视图

3. 调用图

运行时分析工具还能够检测和显示性能瓶颈。这种方法与传统方式相比，最大的优势就是可以获得完美的执行路径概况和关于调用数准确信息。如图 7-11 和图 7-12 所示，Rational Quantify 中的 Call Graph 强调了最大时间消耗执行的路径的调用链，那就是性能弱点。连接方法的粗线，表示了调用链所花费时间和剩余应用时间（如果使用 Purify，那就是内存）的比率。

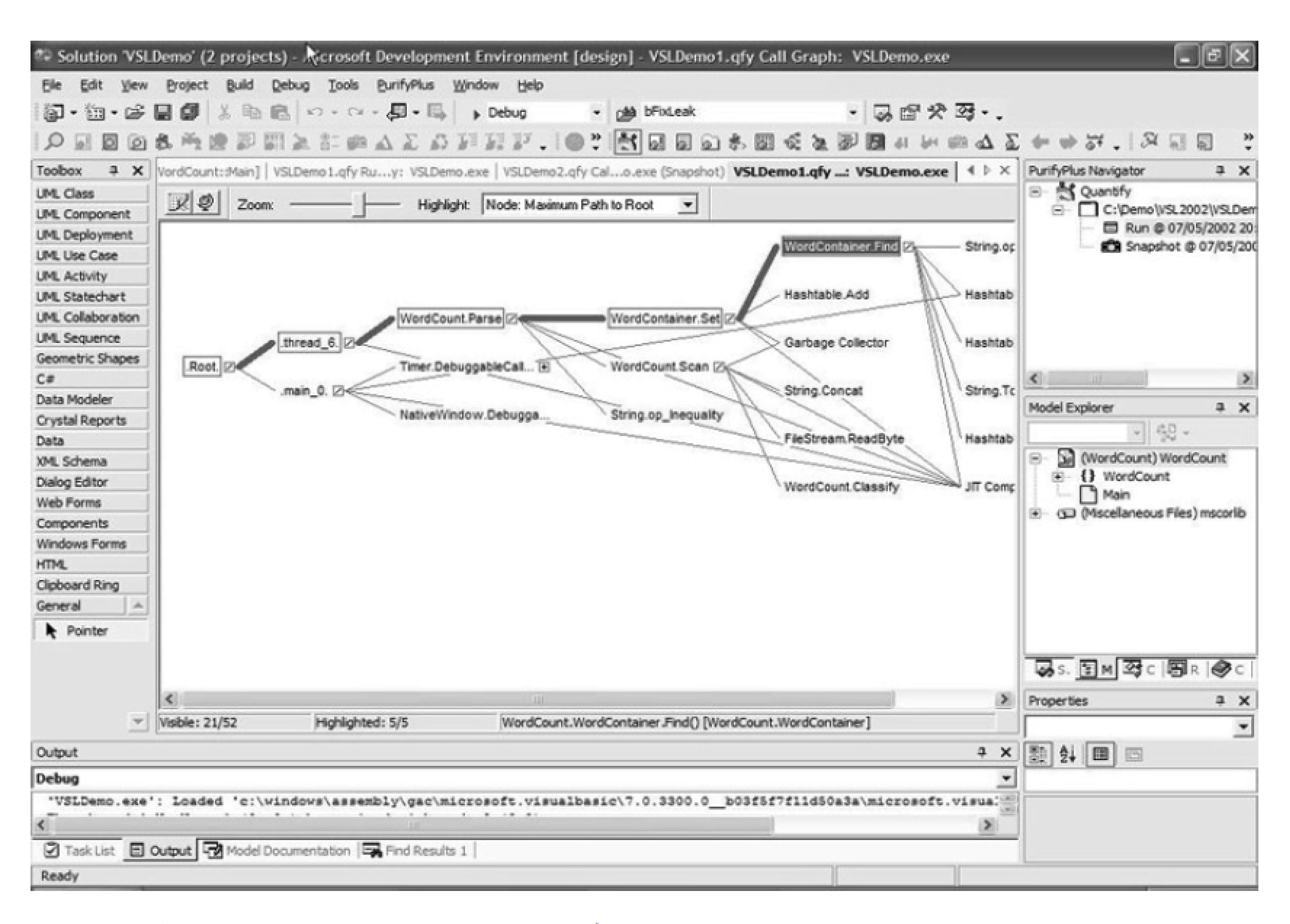


图 7-11　Visual Studio. NET 中 Visual B. NET 和 Visual C＃. NET 应用的混合 Rational Quantify 调用

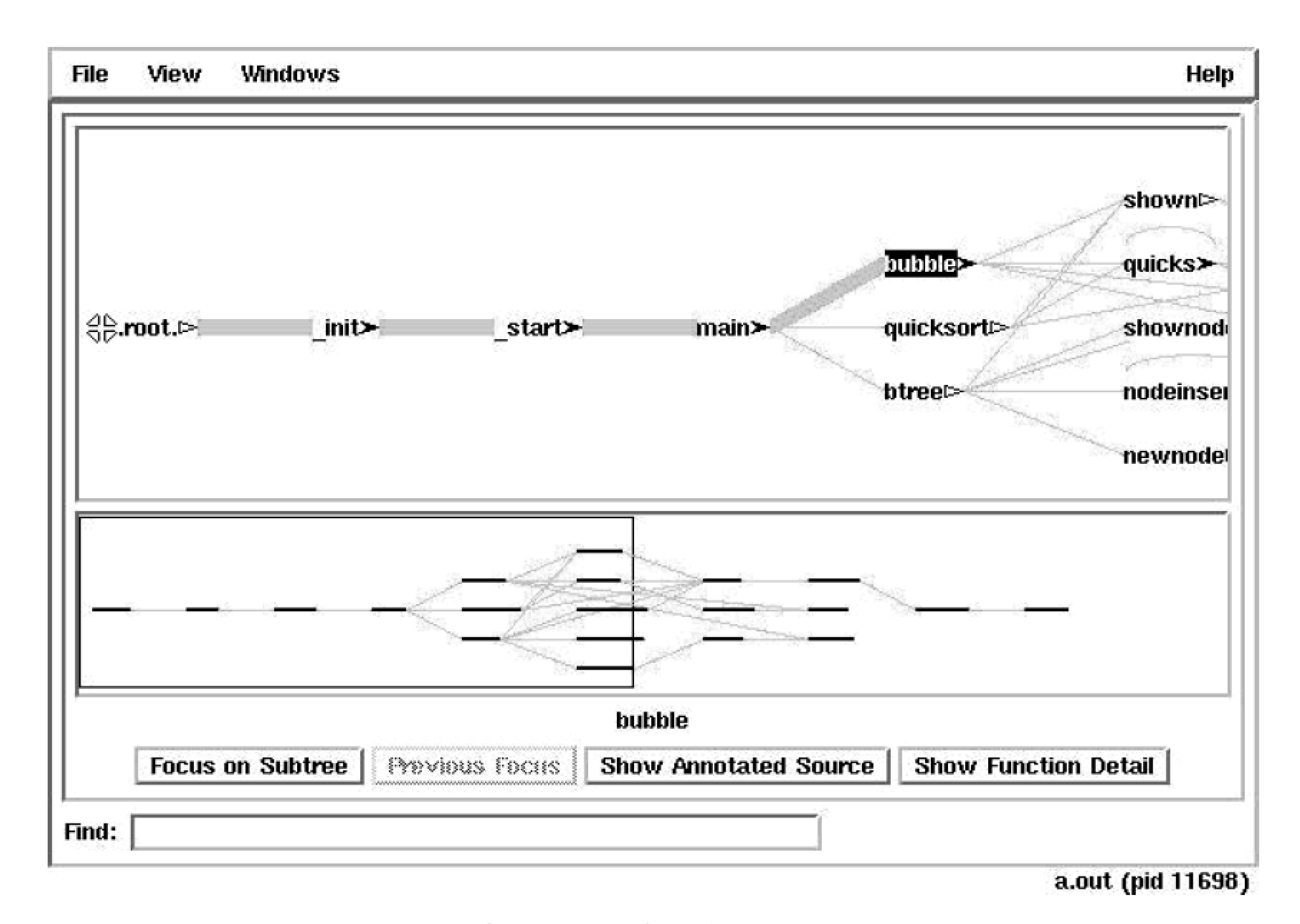


图 7-12　Solaris 中 C/C++ 应用的 Rational Quantify 调用

4. 内存使用

处理内存泄露的第一步就是要检测。一种直观的方式就是可视化全部内存使用，并且在测试下进行内存快照。这样能在运行的应用中发现潜在的内存泄露。例如，如果对

于在服务器端运行的组建内存快照显示，全部内存性能在客户端增加之后，很可能是组件泄露，如图 7-13 所示。

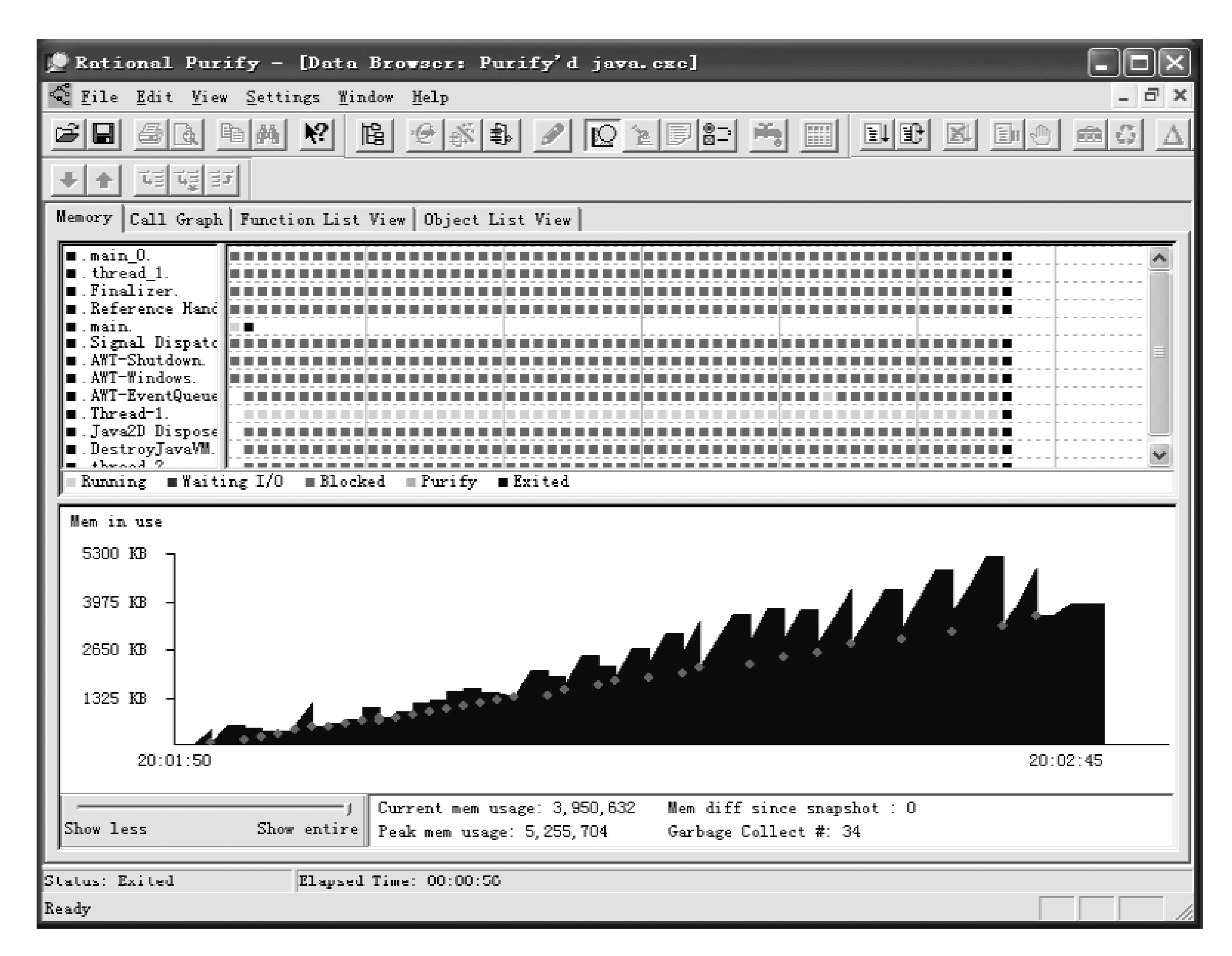


图 7-13　Rational Purify for Windows 中的线程状态和内存使用概况

7.4　组件测试工具

为了保证项目质量和进度的可预见性，就必须要求开发团队对自己开发的代码进行充分测试。但在不借助工具的情况下，开发人员对代码进行完善的测试需要时间相当于整体开发时间的 50%左右，而开发人员的主要职责是开发代码，在面对进度压力时，开发人员进行的测试往往是流于形式，不能切实执行，留下了大量的质量隐患。

这个时候就需要组件测试工具。目前市场上测试工具很多，如前面提到的 JUnit、PurifyPlus 等，都带有部分组件测试功能，但只有 IBM Rational Test RealTime 把组件测试和运行时分析，结合到单一的集成的测试解决方案中，它可帮助开发人员创建测试脚本、执行测试用例和生成测试报告，并提供对被测代码进行静态分析和运行时分析功能。利用该工具，开发人员可以大大提高测试的效率。

7.4.1　Test RealTime 特点

发现并修正缺陷的最佳时间是在开发过程中。这正是 Test RealTime 强调开发人员测试的原因，即只有作为代码的作者才能有效进行那种测试。程序员很容易就能完成测试编写的组件，分析应用程序在运行时的可靠性和性能。该工具包含如下特点：

① 静态分析、功能测试和运行时分析相集成。

② 编辑、测试和调试相集成。

③ Test RealTime 通过分析源代码，自动生成测试驱动(Test Driver)和桩(Test Stub)模板。开发人员只需要在该测试脚本的基础上，指定测试输入数据、期望输出数据以及打桩函数的逻辑。

④ 执行后自动生成测试报告和各种运行时报告。测试报告展示通过或失败的测试用例，而运行时分析报告包括代码覆盖分析报告、内存分析报告、性能分析报告和执行追踪报告。通过 Target Deployment Port 技术同时支持开发机和目标机的测试。

此外，Test RealTime 对 RUP 理论提供了全程支持，利用 Rational Rose 支持 UML 自动建模并提供健壮的代码生成，并能够帮助开发人员，从 UML 模型生成的代码激活运行时分析特性。因此在 IBM Rational 系列工具中，只有 Test RealTime 同时集成了从设计到开发人员测试活动的 UML，提供了业内最广泛的模型驱动开发支持。

7.4.2 开发人员测试现状分析

假设 Test Realtime 软件被安装在 c：\rtrt 目录下，在 c：\rtrt\src 目录下具有 UmtsCode.c 和 UmtsCode.h(通过 winzip 在 c：\目录下展开 rtrt.zip 文件)。其中 UmtsCode.c 中包含了 code_int(int x, char * buffer)函数的实现，该函数的设计规范如下：

① 完成对整数 x 的编码，并把编码的输出值返回到 buffer 中。

② 编码规则如表 7-1 所示。

表 7-1 编码规则

输 入 值	输 出 值
x=2，buffer=""	Buffer "I12" /* 其中 I 表示整数编码，1 为整数串的长度，2 表示整数串 */
x=34，buffer=""	Buffer "I243" /* 其中 I 表示整数编码，2 为整数串的长度，43 表示整数串，进行倒序编码 */
x=56, buffer="I243"	Buffer "I243I265"

对 code_int(int x, char * buffer)进行测试的传统过程如下：

① 利用 C 语言编写测试驱动程序 test_code_int.c，该代码包含 main 函数，并且 main 函数利用输入值调用 code_int，然后检查 code_int 的返回值和期望值是否匹配来判断测试用例是否通过。

② 分别编译 code_int.c 和 test_code_int.c，然后连接执行 test_code_int.exe。

③ 根据 test_code_int.exe 的执行输出，来整理测试报告。

上述传统测试过程存在以下问题：

① 利用 C 语言来编写测试程序，编码工作量大，而且易于出错。测试人员的工作重心不是关注测试用例的设计，而是关注如何实现测试用例。

② 不能对测试程序(test_code_int.c)进行有效的管理，测试执行不方便。

③ 包含测试用例成功与失败的测试报告不能自动化生成，需要手工编写。

④ 不能自动得到代码的覆盖情况，测试完备性和被测试单元的可靠性不能得到保证。



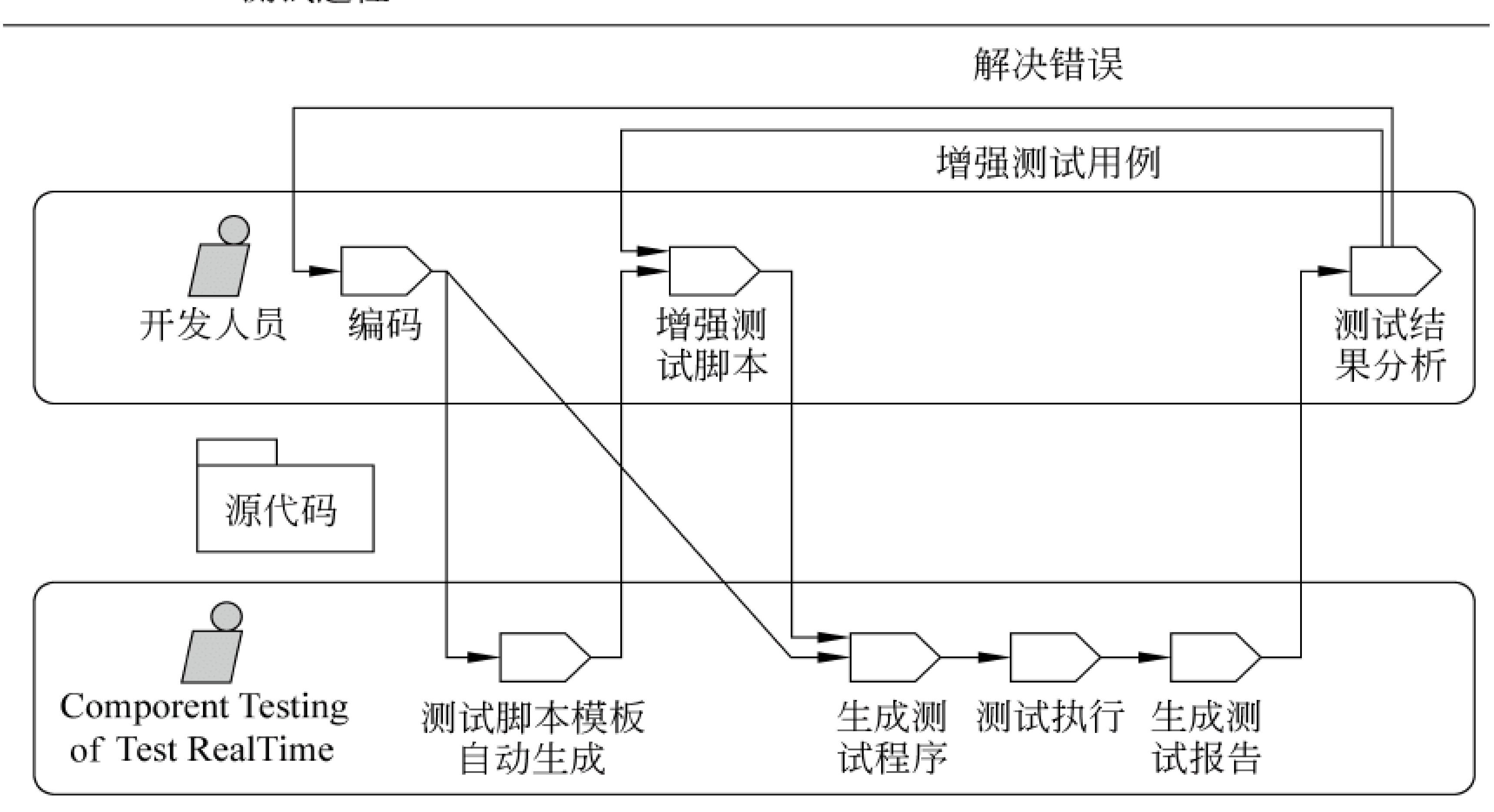


图 7-14　Test RealTime 测试流程

7.4.3　Test RealTime 的开发人员测试过程

图 7-14 是利用 Rational Test RealTime 的开发人员测试过程，步骤如下：

① 编码。开发人员在 Test RealTime 提供的 C/C++ 语言编辑器中进行代码编写。

② 测试脚本模板自动生成。在被测源代码编译通过后，Test RealTime 将通过对源代码进行分析，形成测试脚本模板。

③ 增强测试脚本。开发人员根据设计的测试用例，在测试脚本模板的基础上增加和修改测试用例。

④ 生成测试程序。Test RealTime 将根据测试脚本生成 C 语言测试程序。

⑤ 执行测试。Test RealTime 编译测试程序、被测程序，连接并执行可执行程序。

⑥ 生成测试报告。Test RealTime 将根据测试执行产生的日志文件生成测试报告。

⑦ 测试结果分析。开发人员根据测试报告判断被测程序质量或测试完备性。

⑧ 解决错误。如果发现测试用例未通过，定位错误位置并修改错误。Test RealTime 可以和开发环境的调试器(如 Visual C 6.0 的 msdev.exe)集成，提高错误定位速度。

⑨ 增强测试用例。增强测试用例来覆盖前次测试执行没覆盖的代码分支。

下面以 code_int 的测试为例详细介绍如何利用 Test RealTime 进行测试的过程。

1. 根据源代码自动生成测试脚本模板

选择 File→New→New Activity→Component Testing 菜单，进入 Component

Testing Wizard 界面，通过单击“新建”按钮增加被测文件 UtmsCode.c，并选中 Compute static metrics 复选框对被测代码进行静态分析，如图 7-15 显示。

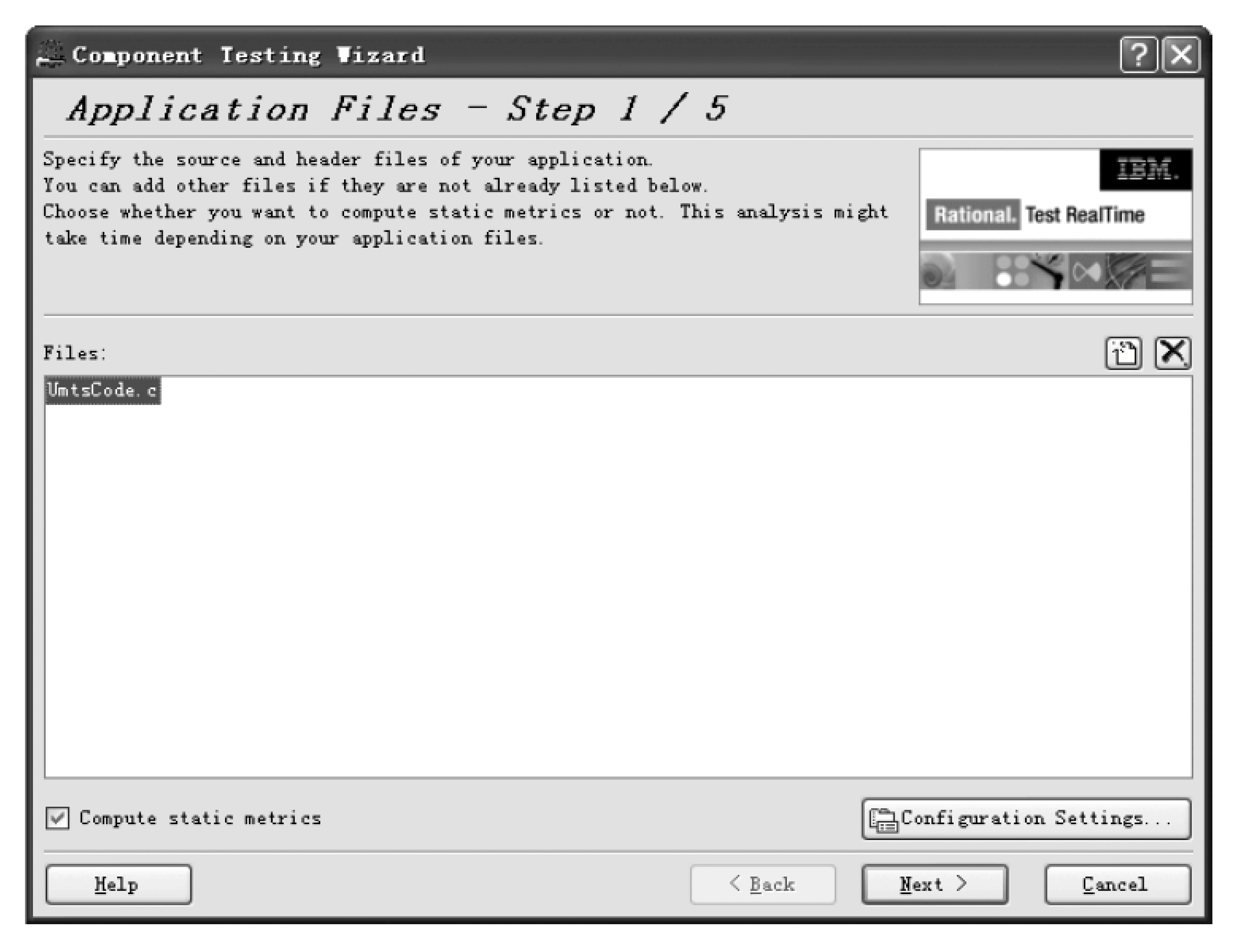


图 7-15　对被测代码进行静态分析

单击 Next 按钮进入如图 7-16 所示的 Components Under Test 界面，选择被测函数。

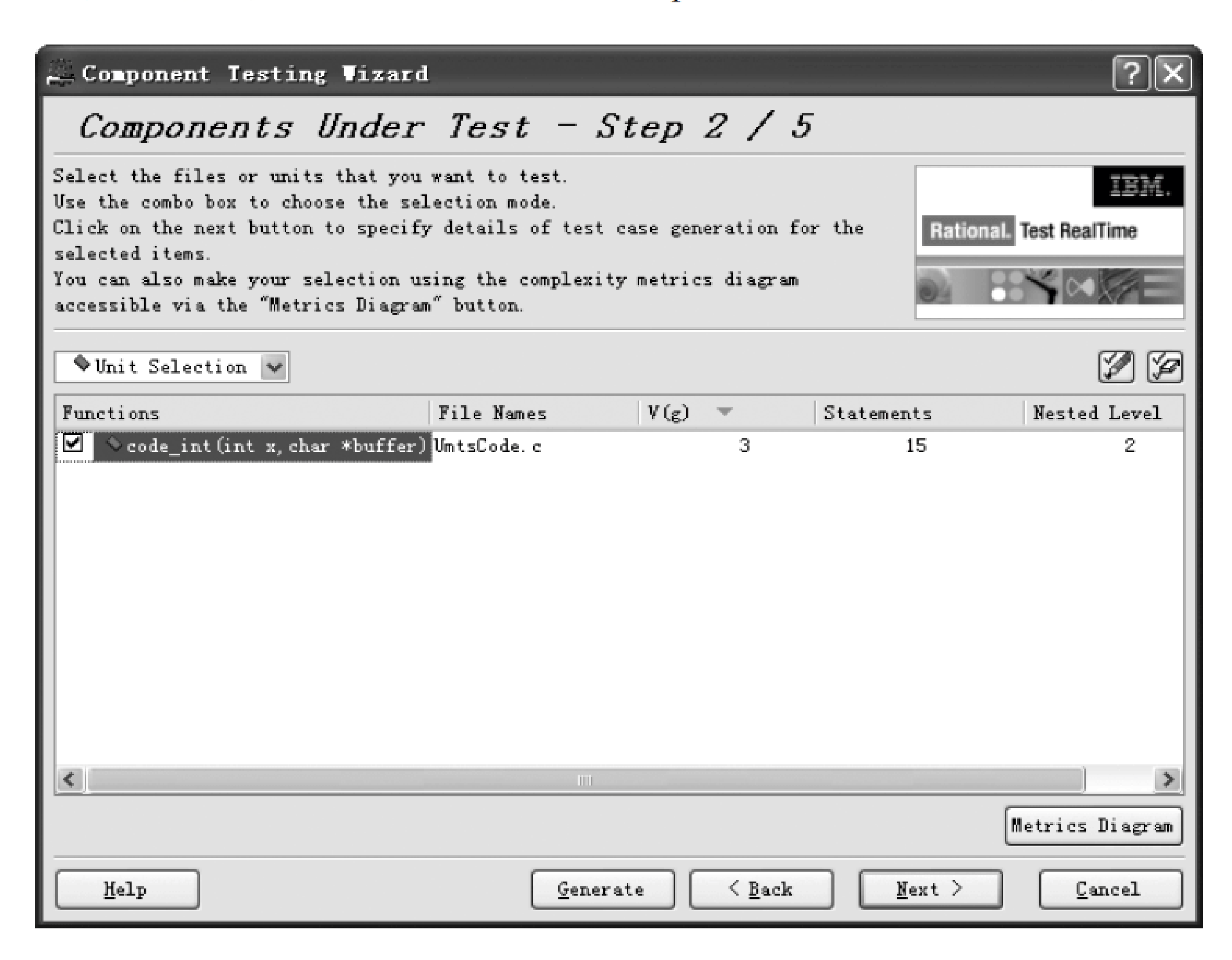


图 7-16　选择被测函数

对于 code_int 函数，v(g)表示与测试难度相关的函数复杂度度量。如 v(g)=1，表示该函数没有分支。为了控制软件的可测试性，建议一个函数的复杂度不超过 10。关于

v(g)的详细解释,参见 Test RealTime 帮助。为了对 code_int 进行测试,选中 code_int 旁边的复选框,单击 Next 按钮进入 Test Script Generation Settings 界面。为了让生成的测试脚本位于 scripts 目录,如图 7-17 所示,修改 Test script path and file name 参数值为“..\scripts\UmtsCode.ptu”。

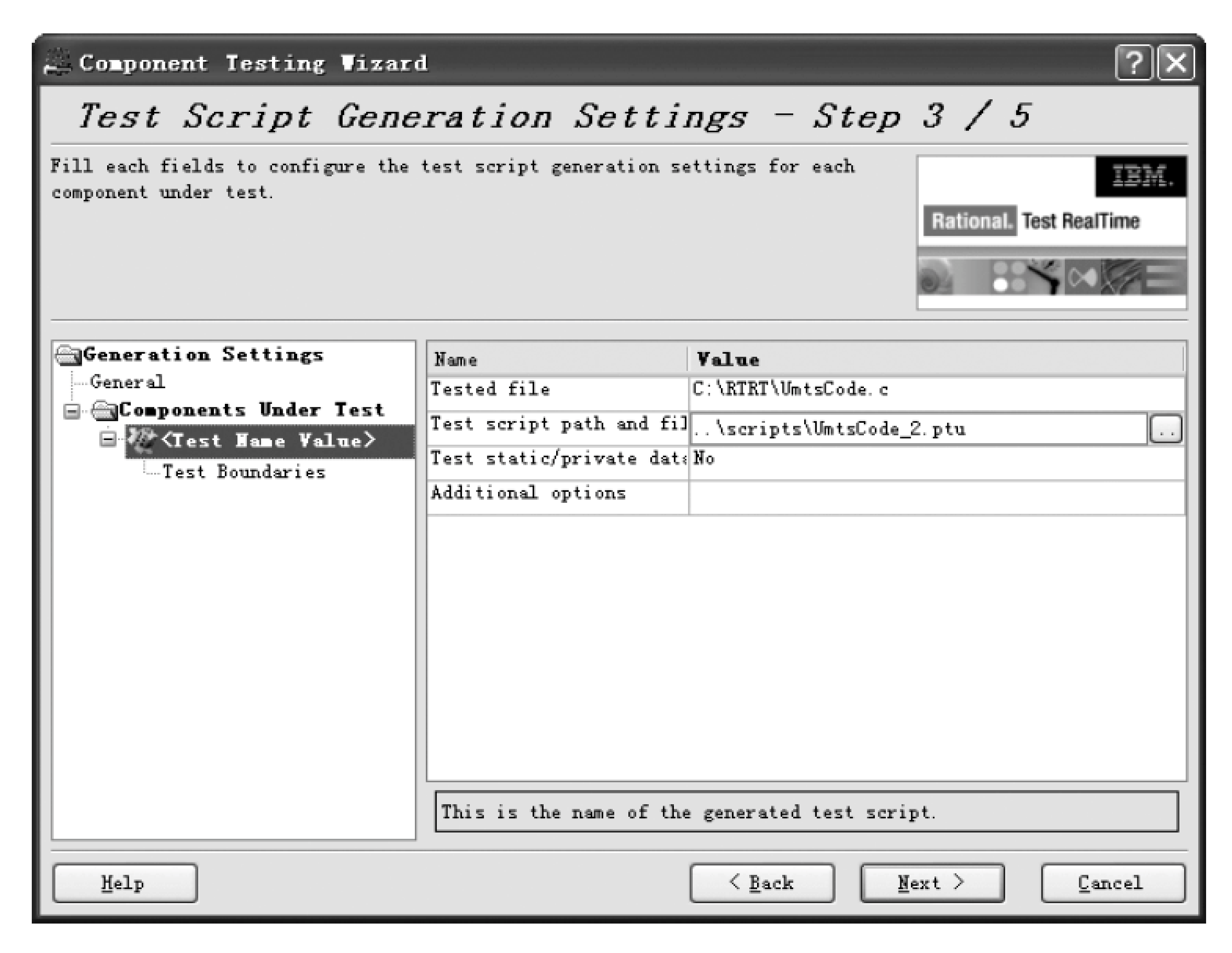


图 7-17 修改参数值

单击 Next 按钮,然后再单击 Finish 按钮,进入如图 7-18 所示的界面。

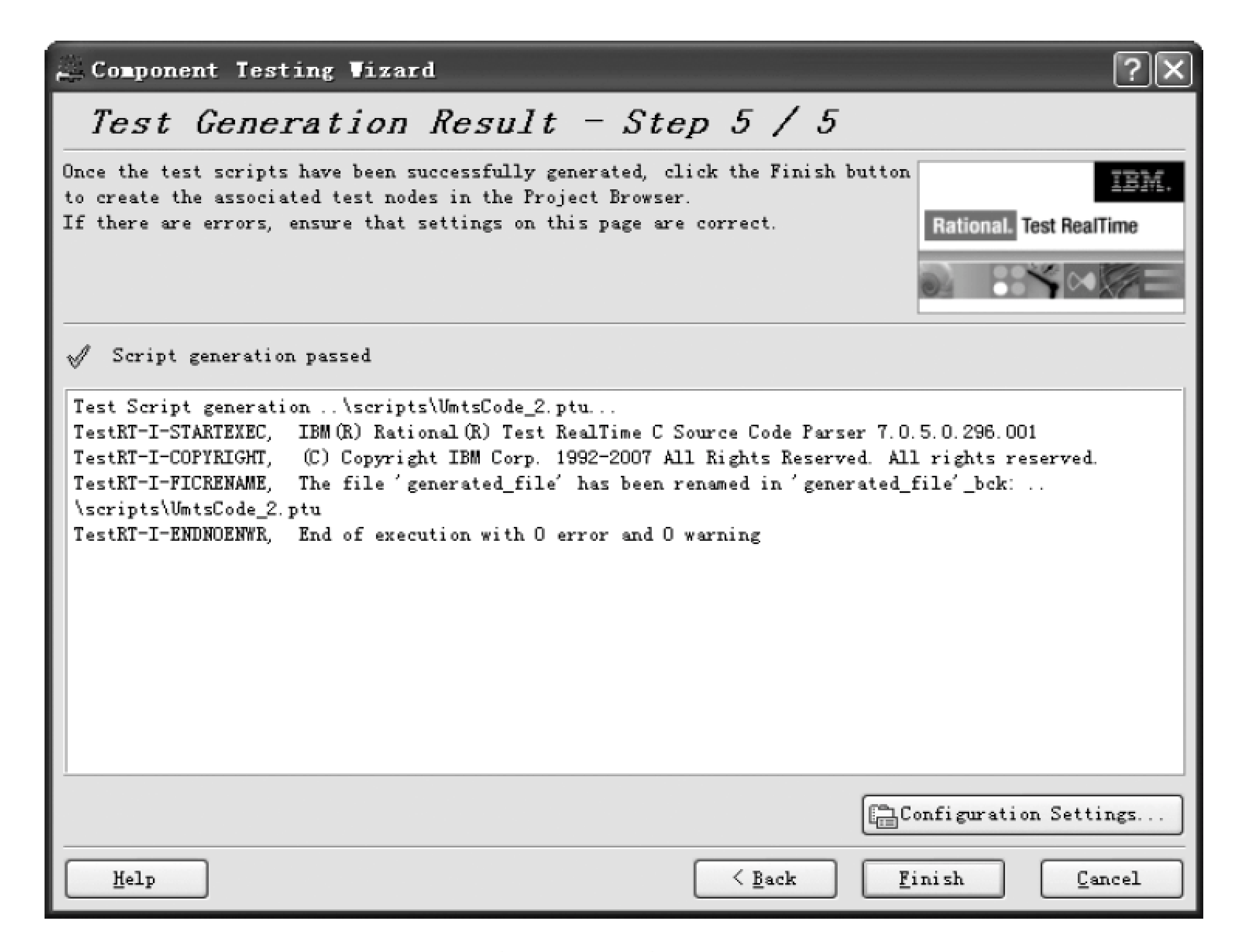


图 7-18 最后界面

上述过程实际是通过图形化界面设置命令 attolstartC 的相关参数。attolstartC 通过分析指定的 C 代码，形成测试脚本模板，详细信息参考 Test RealTime reference manual。

2. 基于测试脚本模板，根据函数的设计规范，编写测试用例

Test RealTime 生成的测试脚本模板中包含一个测试用例，该测试用例的相关输入、输出值设置为 0 或""。

```
SERVICE code_int
SERVICE_TYPE extern
    --Tested service parameters declarations
    #int x;
    #char buffer[200];
    ENVIRONMENT ENV_code_int
    VAR x,          init=0,          ev=init
    VAR buffer,     init="",         ev=init
    END ENVIRONMENT--ENV_code_int
    USE ENV_code_int
    TEST 1
    FAMILY nominal
        ELEMENT
        #code_int(x, buffer);
        END ELEMENT
    END TEST--TEST 1
END SERVICE--code_int
```

其中，"VAR x，init=0，ev=init"语句表示 x 的初始值为 0，期望值等于初始值；"VAR buffer，init=""，ev=init"语句表示 buffer 的初始值为""，期望值也等于初始值。

根据前面 code_int 函数的设计规范，形成如下三个测试用例：

```
SERVICE code_int
SERVICE_TYPE extern
    --Tested service parameters declarations
    #int x;
    #char  buffer[200];
    ENVIRONMENT ENV_code_int
    VAR x,                  init=0,               ev=init
    VAR buffer,             init="",              ev=init
    END ENVIRONMENT--ENV_code_int
    USE ENV_code_int
    TEST 1
    FAMILY nominal
        ELEMENT
            VAR x,          init=2,               ev=init
```

```
            VAR buffer,       init="",                  ev="I12"
        #code_int(x, buffer);
        END ELEMENT
    END TEST--TEST 1
    TEST 2
    FAMILY nominal
        ELEMENT
            VAR x,            init=34,                  ev=init
            VAR buffer,       init="",                  ev="I243"
        #code_int(x, buffer);
        END ELEMENT
    END TEST--TEST 2
    TEST 3
    FAMILY nominal
        ELEMENT
            VAR x,            init=56,                  ev=init
            VAR buffer,       init="I243",              ev="I243I265"
        #code_int(x, buffer);
        END ELEMENT
    END TEST--TEST 3
END SERVICE--code_int
```

3. 执行测试

在修改 UtmsCode.put 测试脚本后，如图 7-19 所示，选择菜单命令 Build 执行测试。

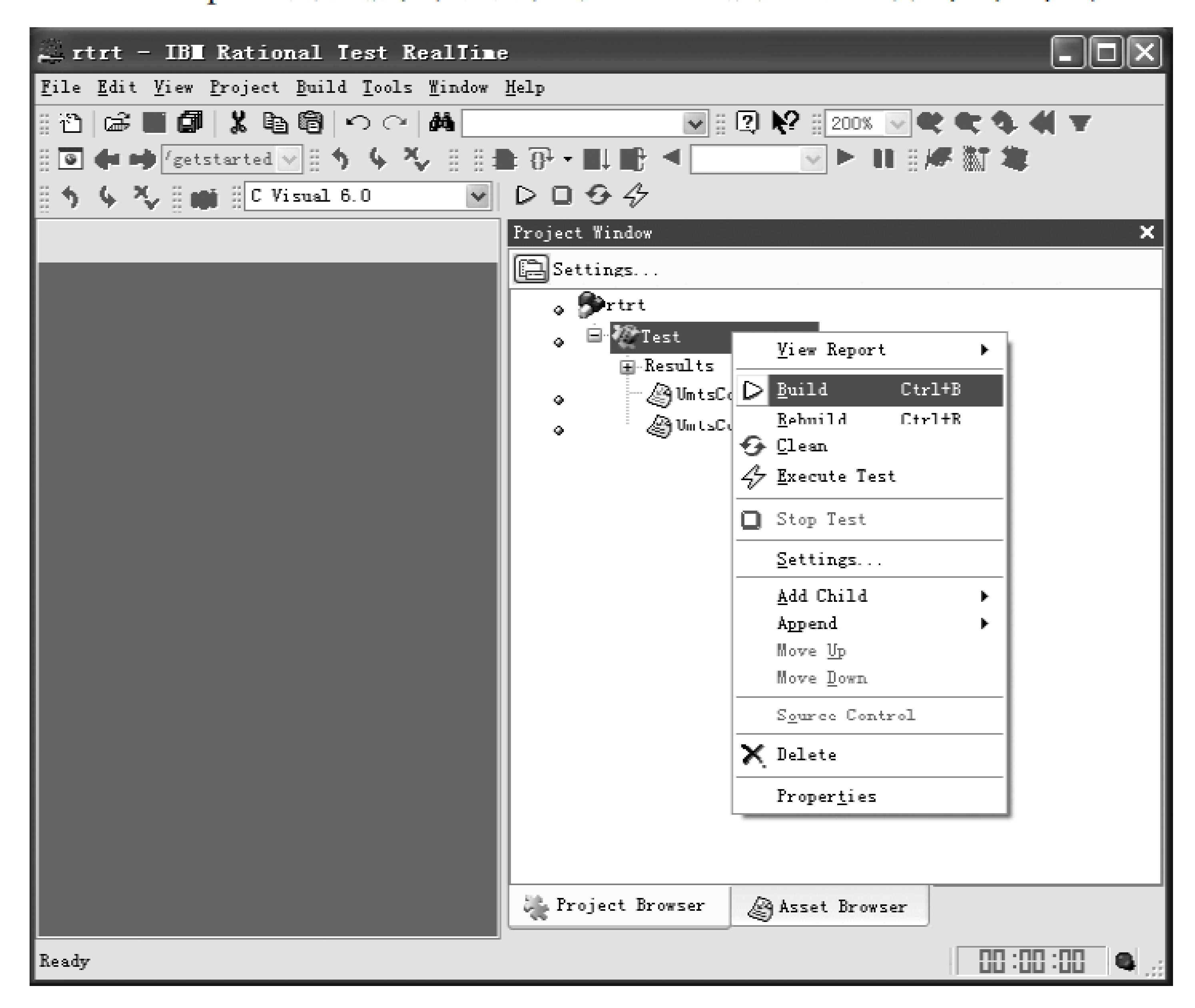


图 7-19 执行测试

在 Build 命令执行过程中，将在输出窗口中显示 build 的详细步骤：

① 执行 attolpreproC 命令把测试脚本 UtmsCode. put 编译成 TTest. c：attolpreproC "C:\rtrt\scripts\UmtsCode. ptu" "cvisual6\TTest. c"。

② 对 TTest. c 进行预处理，编译形成 TTest. obj。

③ 对 UtmsCode. c 进行预处理形成 UtmsCode. i：cl. exe -P "C:\rtrt\src\UmtsCode. c" "-I. . \src。

④ attolcc1 对 UmtsCode. i 进行插针形成 UmtsCode_aug. c：\attolcc1 "cvisual6\UmtsCode. i" "cvisual6\UmtsCode_aug. c" atct. def…。

⑤ cl. exe 编译 UmtsCode_aug. c 形成 UmtsCode. obj：cl. exe -ZI -Yd -GZ -GX -c "cvisual6\UmtsCode_aug. c" -Fo"cvisual6\UmtsCode. obj" "-I. . \src"。

⑥ 计算被测代码 UmtsCode. c 的 Metric：attolstartC "C:\rtrt\src\UmtsCode. c" … -METRICS=". . /reports"。

⑦ 连接形成 Test. exe：link. exe /debug /subsystem:console /machine:I386 /pdb:none "C:\rtrt\test\cvisual6\TTest. obj" "C:\rtrt\test\cvisual6\UmtsCode. obj" "cvisual6\TP. obj" ws2_32. lib /out:". \cvisual6\Test. exe"。其中，TP. obj 是 Test RealTime 提供的库文件。

⑧ 执行 Test. exe。

⑨ 形成测试报告。

在执行过程中，Test RealTime 将以 UML Sequence Diagram 的形式显示被测程序的调用关系，如图 7-20 所示。

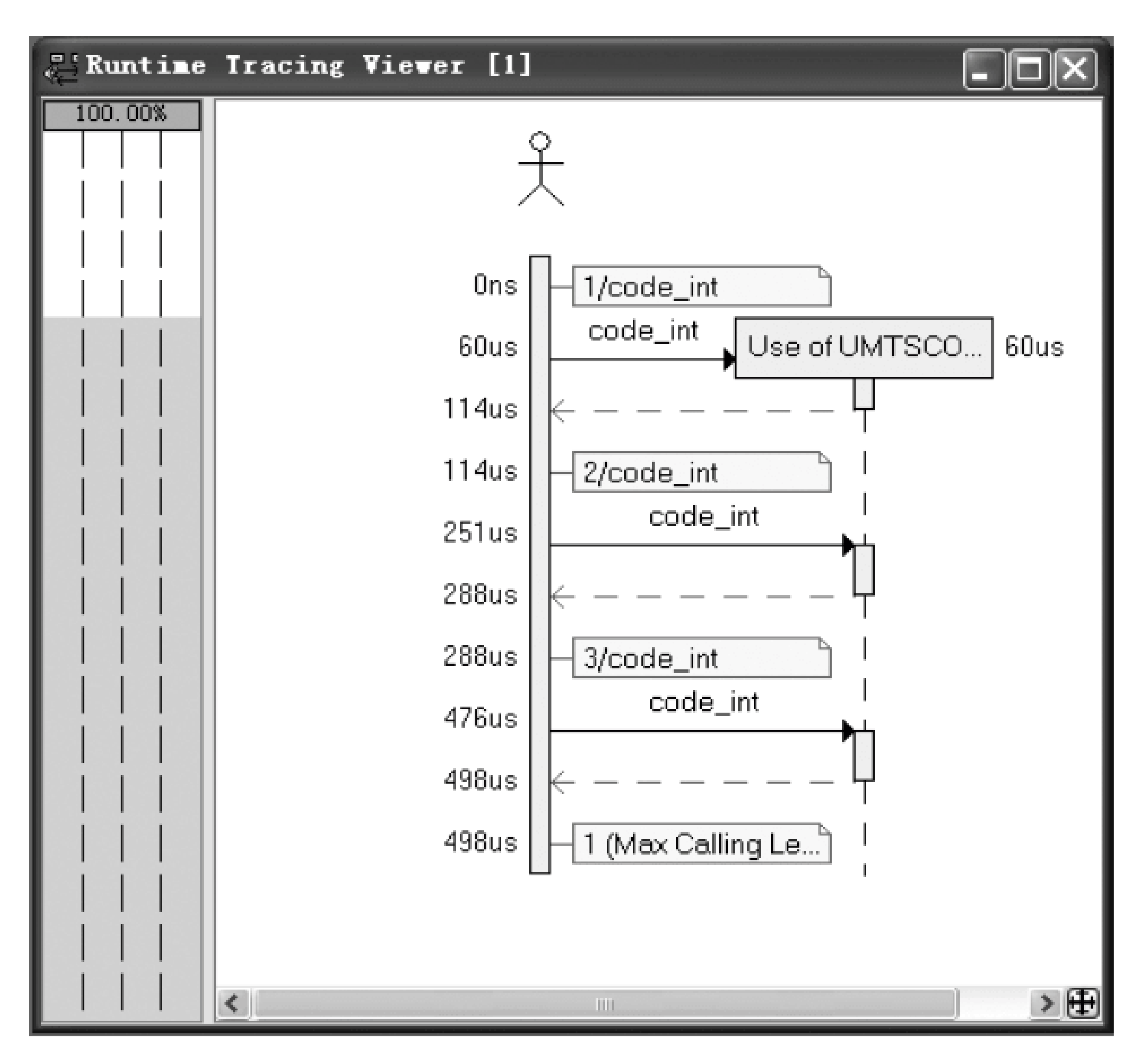


图 7-20　显示被测程序的调用关系

4. 测试结果分析

测试执行完成后，将在 Project Browser 中显示所有的测试报告文件，如图 7-21 所示，这些文件均位于 c:\rtrt\reports 目录。

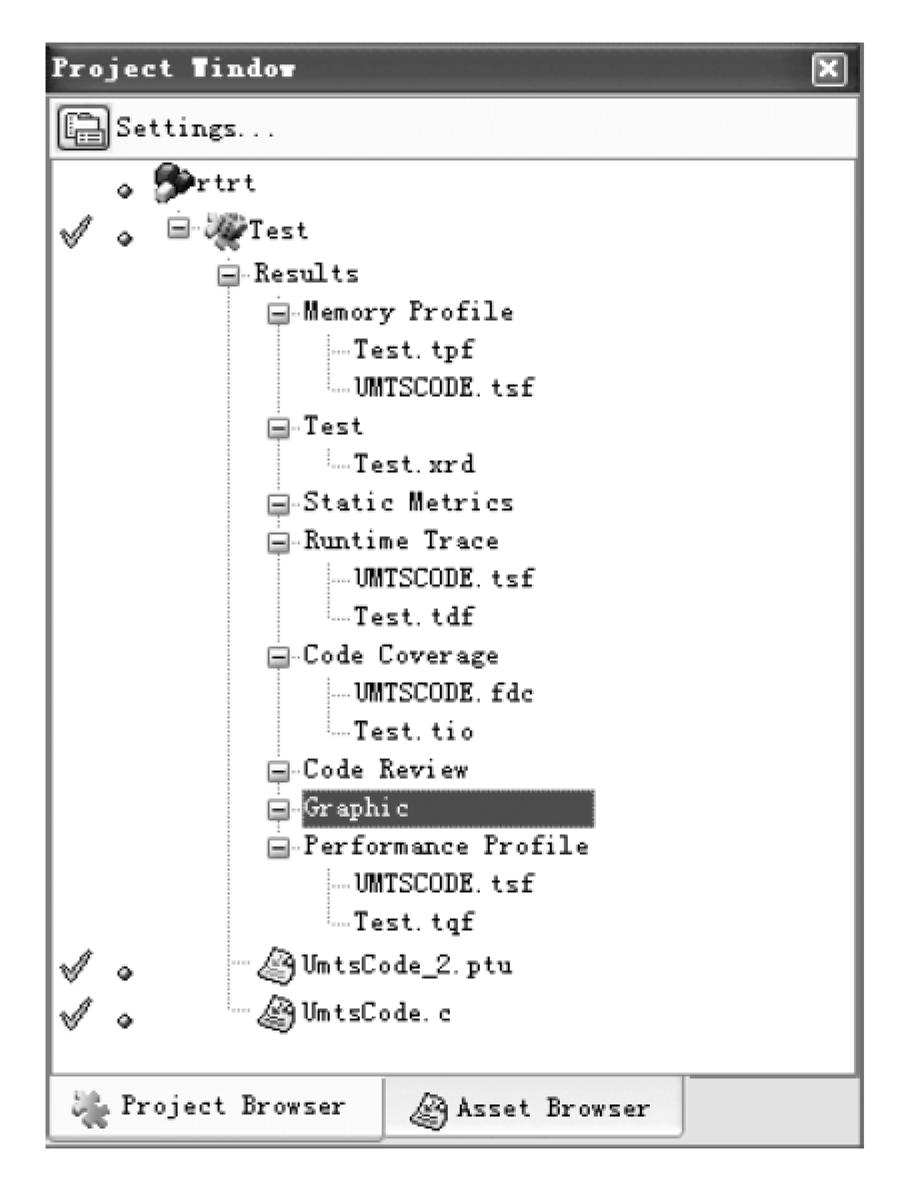


图 7-21 测试结果分析

选中 Results 下的 Test，单击鼠标右键，选择 View Report 命令，进入测试如图 7-22 所示的测试报告，该报告将显示测试用例通过和失败的情况。

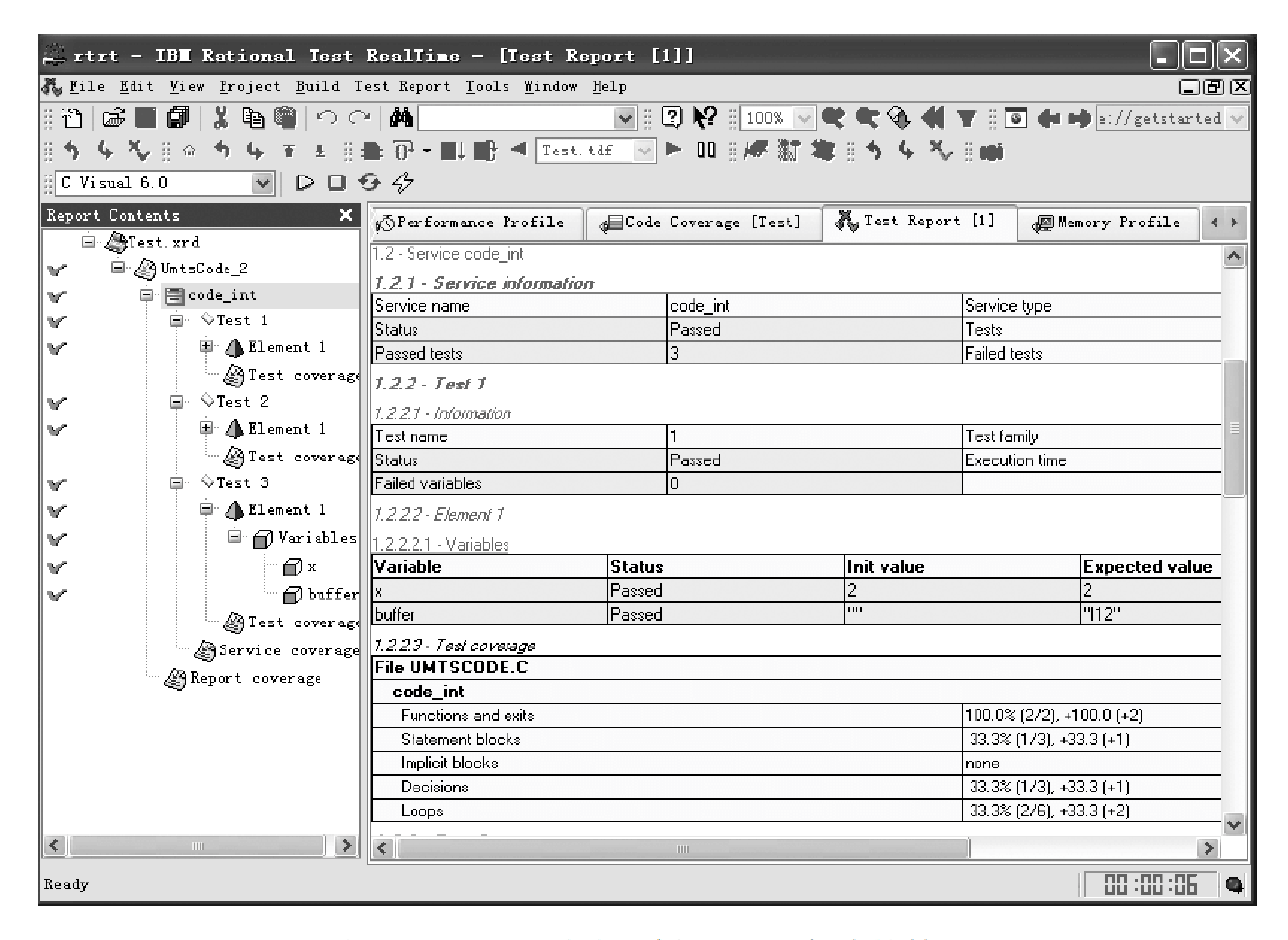


图 7-22 显示测试用例通过和失败的情况

虽然 UmtsCode 函数的三个测试用例都通过，但需要通过代码覆盖情况来分析测试的完备性，因为没有被测试的代码很有可能含有错误。鼠标选中 Results 下的 Code Coverage，单击鼠标右键，选择 View Report 命令，进入测试代码覆盖情况，如图 7-23 所示。

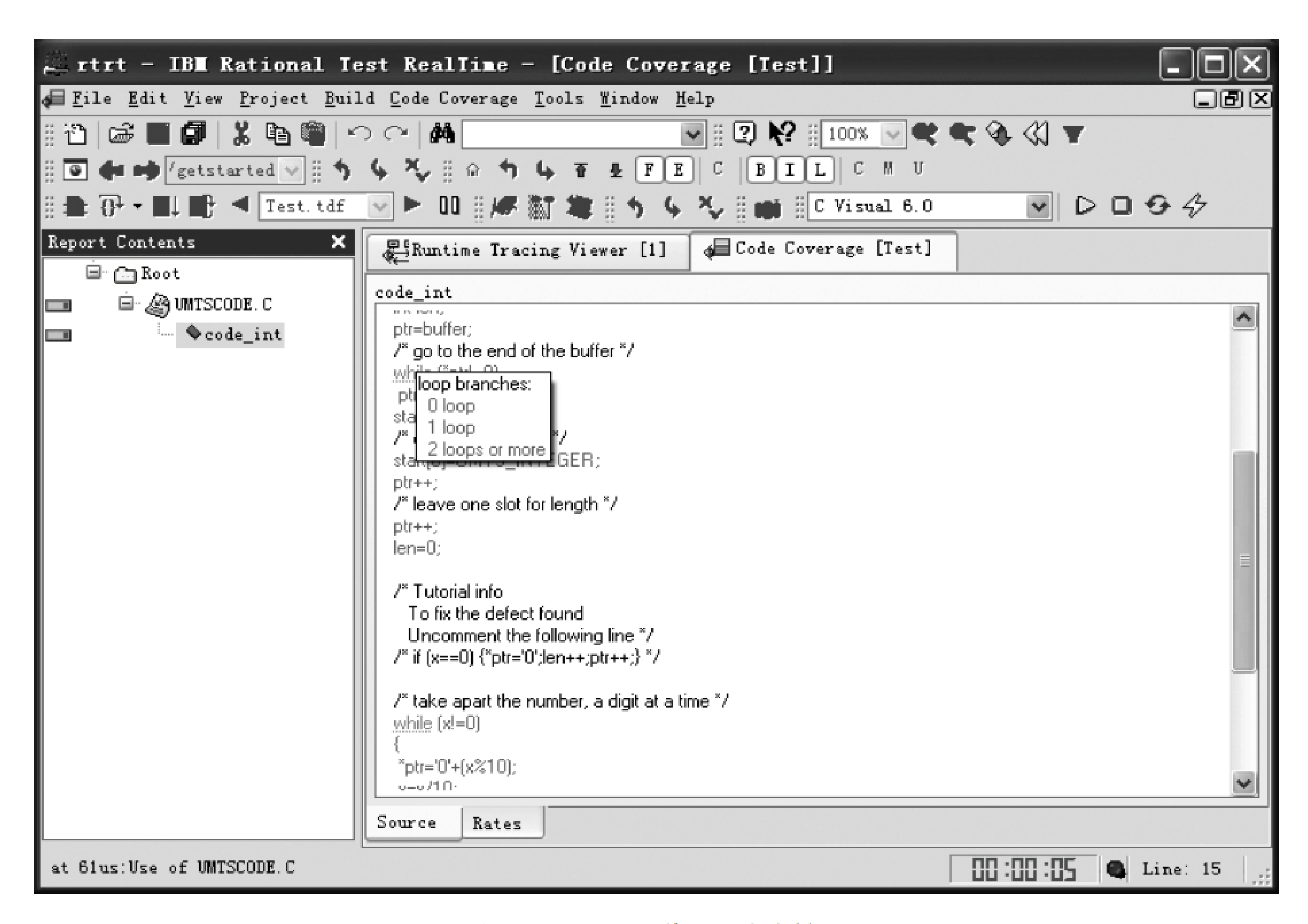


图 7-23　显示代码覆盖情况

在代码覆盖报告中，绿色的代码表示已经覆盖，橘红的代码表示部分覆盖，红色的代码表示没有覆盖。通过对 UmtsCode.c 的代码覆盖情况进行分析，发现 while (x!=0) 语句只是部分覆盖，该语句一次都不执行的这种情况需要完善测试用例。关于代码覆盖的详细信息参考在线帮助。

5. 增强测试脚本 UmtsCode.ptu

通过对代码覆盖情况进行分析，为了覆盖 while (x! =0)的所有情况，需要增加如下测试用例：

```
TEST 4
FAMILY nominal
    ELEMENT
        VAR x,              init=0,             ev=init
        VAR buffer,         init="A",           ev="AI10"
    #code_int(x, buffer);
    END ELEMENT
END TEST--TEST 4
```

增强后的测试脚本参考 c:\rtrt\scripts\ UmtsCode_new2. ptu。再次执行测试，测试报告如图 7-24 所示，发现 Test 4 不通过，表明 UmtsCode. c 中有错误。

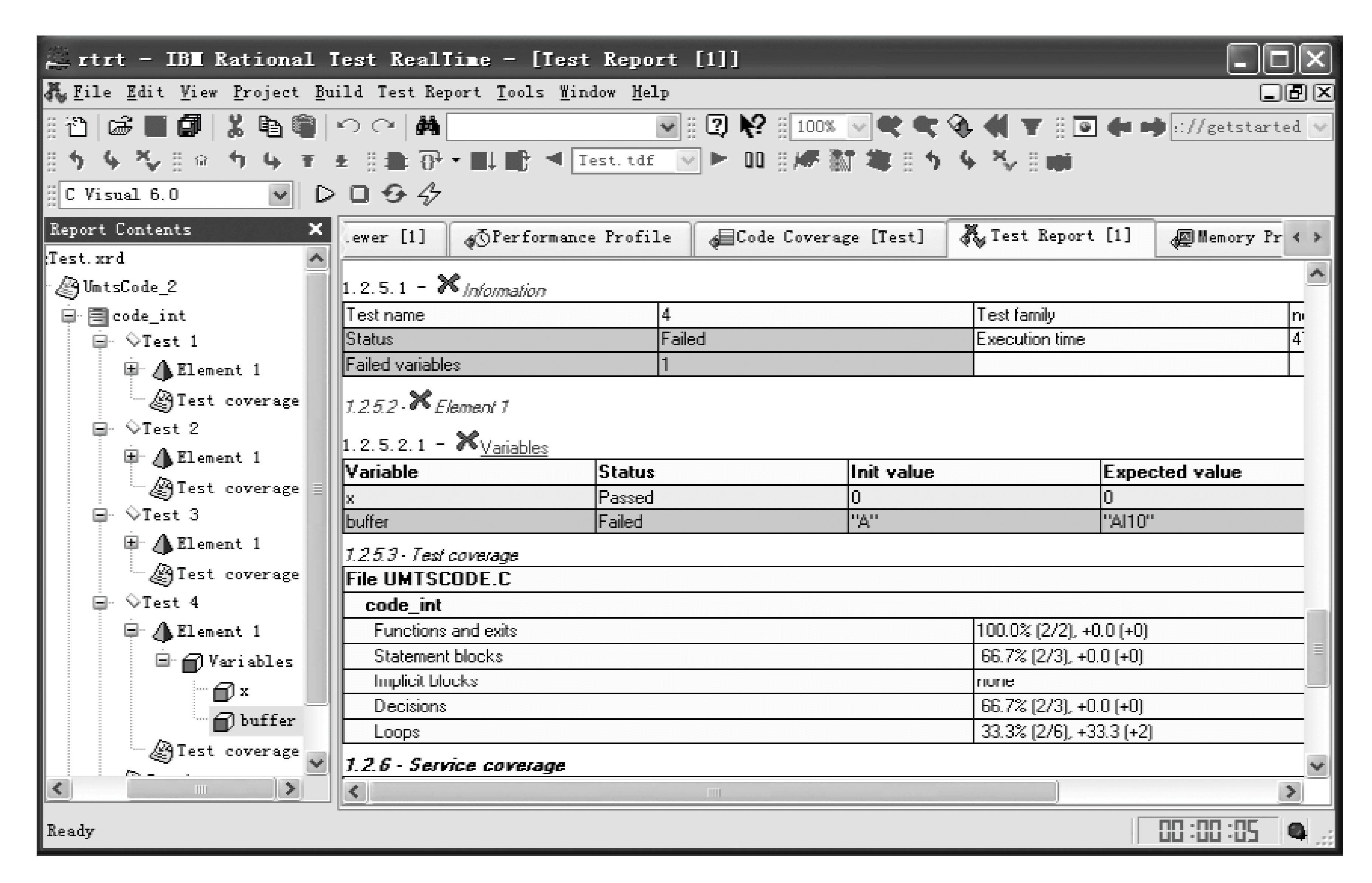


图 7-24　显示测试报告

6. 修改测试脚本 UmtsCode. c

如附件 UmtsCode_new. c 的内容修改 UmtsCode. c，然后重新执行测试，就可以发现测试报告中的所有测试用例都已通过，同时所有的代码均已被执行。

7.5　总　　结

组件系统的集成测试对于保证系统软件质量是非常重要的。本章介绍了组件技术的由来，解释了组件是系统中一个有价值的、近乎独立的可替换部分，要具备可重用性和可替代性。同时对组件的特点、三个流派、形态进行了说明。重点介绍了 UML 对组件测试的意义，以及组件测试的方法和特点。

接着还介绍了运行时分析技术，运行时分析是通过在调试器中运行目标程序，观察执行过程中的代码，以发现潜在问题。本章解释了它与静态分析的不同。并对运行时分析进行了分类，详细说明了运行时分析的两种方式，即关键运行时参数的测量和运行时行为的文档，还给出了运行时分析的具体案例。

最后介绍了组件测试工具 IBM Rational Test Realtime，Test RealTime 除了支持 C 语言外，还支持 C++ 和 Java 语言，并支持基于消息的测试。开发人员利用该工具，可实现编码、测试和调试的有机集成，使得边开发边测试或测试驱动的开发得以切实执行。还介绍了 Test RealTime 的一些基本的用法、工具的特点、开发人员测试现状分析、开发人

员测试过程等。

习题与思考

1. 什么是组件?
2. 为什么要进行组件测试?
3. 组件有哪些特点?
4. 组件测试为什么要引入 UML?
5. 组件测试有哪些方法?
6. 什么叫运行时分析?
7. 运行时分析有哪两种工作方式?
8. 运行时分析与静态分析有什么区别?
9. 简述 Test RealTime 的特点。
10. 叙述 Rational Test RealTime 的测试过程。

第 4 部分

系统测试

工程师所面临的问题是修改软件时会引起系统混乱，特别是一个微小的错误就能导致系统崩溃。但是，修改也能带来机遇。简而言之：如果很轻易地就能给系统增加一定功能，那么就会冒一定的风险，才能增加更多的功能。从而使我们的计划显得有些疯狂——我们将倾向于尽可能地冒风险。

——美国测试专家 James Bach

James Bach 的话说明了修改软件增加功能而存在风险，而系统测试的重要性，就在于尽可能降低增加功能而带来的风险。

RUP 指出当把软件作为整体运行或实施明确定义的软件行为子集时，即可进行系统测试。这种情况下的目标是系统的整个实施模型。系统测试应该按照测试计划进行，其输入、输出和其他动态运行行为，应该和软件规约进行对比。通过系统测试可以发现很多软件缺陷，例如子系统间的交互、运行条件、内存使用等，而这些是软件开发人员发现不了的。

本部分包括第 8 章和第 9 章，读者可以了解软件系统测试方法，主要有功能测试、性能测试等。

第8章 系统功能测试

小故事：Intel 浮点除法软件缺陷

Intel 公司生产的 CPU 以质量优良、性能优异闻名于世，但 Intel 历史上曾遭遇过最大的公关危机。1994 年 12 月弗吉尼亚州 Lynchburg 大学 Thomas R. Nicely 博士在 Pentium PC 上做除法实验时发现了异常，这个问题就是计算(4 195 835/3 145 727)×3 145 727—4 195 835 的答案本来是 0，但 CPU 显示结果却是其他数。他把发现的这个问题放到了 Internet，引发了一场争议风暴。事情越闹越大，最终 Intel 为没有及时解决软件缺陷而道歉，并支付 4 亿美金来更换芯片。事后发现原因是英特尔为了加速运算，将整个乘法表烧录在处理器上面，但是 2048 个乘法数字中有 5 个输入错误。这显然是系统功能测试出了问题，后来查明：Intel 软件测试工程师在进行功能测试时，已经发现了这个问题，却没有引起测试经理的重视。

软件的功能测试往往被认为是测试中相对简单的工作，缺乏技术含量，只是"简单的鼠标操作"。实际上软件功能测试，一方面依赖于不断积累的经验，另一方面功能测试也是离不开技术，包括环境设置、功能实现的理解。如果结合测试自动化、白盒和黑盒测试方法等，测试的效率会更高。

毫无疑问，严格的功能测试是成功开发应用的关键。开发人员、测试小组和管理人员所面临的挑战，是如何加速测试流程和提高测试的精确性和完备性，同时还不能增加已经很紧张的预算。通过将功能测试的关键环节自动化，可以达到规定的发布时间要求，而且功能测试将更加全面和可靠，保证业务过程功能的正确性，从而从上线的运营中，获得极高的价值和客户满意度。

学习本章，读者需要了解功能测试的定义以及与单元测试的区别。重点掌握正则表达式，还有 Robot、Rational Functional Tester 工具的使用，尤其是对象识别技术、验证点、ScriptAssure 等技术需要理解和掌握。

8.1 什么是系统功能测试

功能测试一般须在完成集成测试后进行，而且是针对应用系统进行测试。功能测试是基于产品功能说明书，在已知产品所应具有的功能，从用户角度来进行功能验证，以确

认每个功能是否都能正常使用，是否实现了产品规格说明书的要求，是否能适当地接收输入数据而产生正确的输出结果等。功能测试包括用户界面测试、各种操作的测试、不同的数据输入、逻辑思路、数据输出和存储等的测试。对于功能测试，针对不同的应用系统，其测试内容的差异很大，但一般都可归为界面、数据、操作、逻辑、接口等。具体包括以下几个方面：

- 程序安装、启动正常，有相应的提示框、适当的错误提示等；
- 每项功能符合实际要求；
- 系统的界面清晰、美观，菜单、按钮操作正常、灵活，能处理一些异常操作；
- 能接受正确的数据输入，对异常数据的输入可以进行提示、容错处理等；
- 数据的输出结果准确，格式清晰，可以保存和读取；
- 功能逻辑清楚，符合使用者习惯；
- 系统的各种状态按照业务流程而变化，并保持稳定；
- 支持各种应用的环境，能配合多种硬件周边设备，与外部应用系统的接口有效；
- 软件升级后，能继续支持旧版本的数据。

软件功能测试主要围绕 Windows 图形界面、字符终端和 Browser 界面进行测试。客户端可以是 VC、VB、PB、Delphi 等编制的软件，各种字符终端软件，或在客户端运行浏览器 Microsoft Explorer 和 Netscape，通过自动录制形成测试脚本，实现自动化的功能/回归测试。

8.1.1 功能测试要素

软件产品以软件的客户为出发点，好的用户界面，除了正确性和实用性之外，还包括另外 5 个要素：符合标准和规范、直观性、一致性、灵活性、舒适性。

① 符合标准和规范。软件在现有的平台上运行，通常标准是已经确立的（如 MAC 或 Windows），这些规则和约定也是功能测试的依据。这些标准和规范是在大量的实践基础上，为方便用户而累积下来的各种规则和约定，如软件菜单格式、快捷键、复选框和单选按钮的界面，使用提示信息、警告信息或严重警告信息等特定信息。

② 直观性。首先了解所需的功能或期待是否响应明显，并在预期的地方出现。其次要考虑用户界面的组织和布局是否合理，界面是否洁净、不拥挤，是否有多余的功能，是否有太复杂难以掌握等因素。

③ 一致性。软件自身的一致性以及软件与其他软件的一致性。字体和界面的各元素风格是否一致是比较容易判定的，而一致性较难的判断体现在用户操作方式上。用户习惯于将某一程序的操作方式带到另一个程序中使用。例如在 Windows 平台客户已经习惯用 Ctrl+C 表示复制操作，而如果在软件中将复制操作的快捷键定义为其他键，必定会给用户造成挫败感，难以接受。

④ 灵活性。软件可以选择不同的状态和方式，完成相应的功能。但灵活性也可能发展为复杂性，太多的状态和方式的选择增加的不仅是用户理解和掌握的困难程度。多种状态之间的转换，增加了编程的难度，更增加了软件测试的工作量。

⑤ 舒适性。人们对舒适的理解各不相同,但总体上要求恰当的表现、合理的组织、色调和谐、必要的提示等。

8.1.2　功能测试的注意事项

在测试前,首先要根据《需求分析报告》全面了解用户需求并透彻理解。测试时要注意以下几点:

① 测试时要分清主次,即先测试主要功能,后测试次要功能。要先找出系统的功能主干,让数据依次流经功能主干,测试功能实现是否正确。如果功能主干有问题,这个系统就是失败的。

② 确定功能主干正确后,还要考虑测试其异常处理功能。

③ 功能主干测试正确后,再进行分支功能的测试。

④ 要对程序的功能进行方便性测试,将不够满意的地方,都应当成系统缺陷向项目负责人或系统开发者指出。

⑤ 检查系统需求和设计说明书中要求的功能是否在系统中都被实现,性能是否达到指标。

⑥ 数据之间的逻辑关系是否正确。

⑦ 要有预览和打印功能。对于企业端软件,打印不能只针对一种打印机,要用多种打印机进行测试。

8.1.3　场景测试

第 2 章已经知道场景是事件触发时的情景,是用例的一个实例。场景技术在软件生命周期中有着广泛的应用。在需求分析阶段,场景可以用来捕获需求和系统的功能;在软件设计阶段,场景是软件体系结构建模的主要依据,它是软件主要行为的集中体现。场景主要用来简要描述系统预期的或所希望的使用方式。

从外部用户的角度看系统的执行过程,反映系统的期望运行方式,场景与白盒测试中的路径很相似,但还是有一定差别。场景的路径更为细致,涉及算法内部的分支情况,但场景并不考虑系统的内部设计细节。场景适合描述系统中的任何角色,包括操作员、系统设计人员、修改人员、系统管理人员和其他人员,他们从事的活动以及他们之间的交流或通信。

实践中,人们常常会从各种不同的特定场景的角度来展示和分析期望系统实现的功能。针对具体的系统和对其特定的需求,场景大多用以描述系统的各种典型的活动过程。在系统的开发过程中,随着需求的不断丰富和设计的逐步细化,通常会有更多的更加典型和具体的场景被确定下来,据此可以不断捕获各种典型的系统应用案例,并进而为测试构造具有代表性的测试用例。

测试场景对于测试人员来说很重要。因为现实中进行系统测试时,必须全面考虑系统的用户(各种角色)使用时会出现的种种情况,包括正常的和非正常的,来考查系统的容

错性、健壮性等。虽然软件的黑盒测试在软件的接口处进行，不考虑程序的内部逻辑结构和逻辑特性，但是实施系统测试时，程序运行的上下文环境总会影响模块的执行轨迹，而最终影响系统的执行结果。因此，有必要在测试时把描述系统的执行过程的场景作为一个参考，以利于保证测试的全面可靠。通过分析，根据需求设计生成场景可以罗列以下三个作用：

① 验证模型设计是否满足需求。需求捕获中最大的问题是软件人员与用户之间的沟通障碍，设计的模型不能准确描述用户的要求或者与初始的想法有些偏离，根据模型自动生成的场景可以反过来验证模型的正确性，保持整个软件开发过程的一致性。

② 为软件走查提供依据。参加走查的评审人员要在很短时间内读懂软件设计模型(如 UML 模型)可能有些困难，而场景可反映系统的期望运行方式，从用户的角度来看，易于被他人理解，从而易于有的放矢地提出问题，减少了走查时的随机性。

③ 为功能测试用例的生成打下基础。一般来说，一个测试用例是指依据某个特定的测试要求设计的，用于驱动被测软件。从某一状态(如初始状态)执行到另一个预定状态(如终止状态)所需的外部设置、操作指令和输入数据的集合。在大型复杂软件的测试实践中，特别是对于分布式和交互式系统的测试，测试用例通常包括一个特定的测试场景和与之相对应的一组输入数据，包括操作指令、输入数值和初始化设置值两个部分，所以场景的生成是必需的。

8.1.4　功能测试与单元测试的区别

单元测试好比房屋建筑现场的建筑监理员，他关心房屋的各个内部系统，如地基、构架、供电系统和管道设备等。房屋每部分工作是否都安全、正常。单元测试是从开发者的角度来编写的。它们确保类的每个特定方法成功执行一系列特定的任务。每一个测试都要保证对于给定的一个已知的输入应该得到所期望的输出。

功能测试类似于观察同一建筑现场的房主，他假定内部系统将正常运作，并假定建筑监理员在执行其任务。房主关心的是住在这所房子里将会怎样。他关心房子的外观如何，各个房间的大小面积是否合适，房子能否满足家庭生活的需要，以及窗户的位置是否有利于采光等。

UAT(User Acceptance Test，用户接受度测试)主要采纳场景测试(Scenario Test)，场景测试关注于不同场景、事务、业务流程等，仅用到各个功能的一部分处理流程；一个场景测试用例仅测试一个场景、事务或业务流程。

三者的关系：房主对房子执行功能测试，是从使用的角度考虑问题。建筑监理员对房子执行单元测试，是从建筑细节质量的角度考虑问题。功能测试是场景测试的先决条件，只有功能测试已经完成，且发现的问题得到了解决，场景测试才可能有效地得到实施；如果在场景测试中发现了大量本应在功能测试中发现的问题，那么说明功能测试急需加强。

功能测试是单元测试的补充，但又有很大不同。简言之，单元测试说明了代码执行是

否正确；功能测试看完成的功能是否正确。单元测试往往是从代码开发人员的角度来看问题，而功能测试是从最终用户和业务过程角度来看问题。

8.2 Web 功能测试

随着网络技术的迅速发展，各类基于 Web 的应用程序以其方便、快速、易操作等特点成为软件开发的重点。以 Web 为核心的应用日益广泛，越来越多的公司采用 B/S 模式开发其电子商务网站。在众多基于 Web 的应用程序开发过程中，如何有效地进行系统测试，以提高 Web 应用软件的正确性、有效性、可靠性，成为研究的重要课题。

由于 Web 应用程序的开发过程需要综合 HTML、JavaScript、VBScript、数据库技术、网络技术等多种技术，所以对 Web 应用程序的测试与传统的软件测试技术有很大区别，下文针对 Web 功能测试技术的多方面进行讨论。

根据功能测试用例，逐项测试，检查产品是否达到用户的要求。Web 应用程序由多个页面链接组成，页面中往往包括大量表单，并涉及数据库的应用。在具体实现过程中采用 Cookies 保存用户信息。所以 Web 功能测试主要包括链接测试、表单测试、Cookies 测试、数据库测试和设计语言测试等方面。

1. 链接测试

超链接是 Web 应用系统的一个主要特征，一个 Web 应用程序由多个页面组成，页面和页面之间通过链接进行跳转。一个完整的 Web 应用系统应该保证没有无效链接、错误链接和孤立页面。其中无效链接是指链接的页面不存在，错误链接是指没有按指示的内容链接到指定页面，孤立页面指没有链接指向、只有知道正确的 URL 地址才能访问的页面。

链接测试在集成测试阶段完成，在 Web 应用系统的所有页面开发完成之后进行。以往的链接测试多采用手工测试，这种方法对小型 Web 应用程序有一定作用，但是如果 Web 应用程序比较大，包含的页面过多，则难以通过手工点击的方法找出错误。现在已有一些链接测试的工具可以采用，例如 IBM Rational Appscan 等。

2. 表单测试

在 Web 应用系统中，通过大量表单提交信息。所以在功能测试中，必须测试表单中提交的信息的正确性、完整性。例如，表单内容是否为空，身份证号、邮政编码、手机号码的位数是否正确，E-mail 地址格式是否正确，密码是否符合安全规则等。要保证这些完整性，需要编写表单验证程序。提交表单时调用表单验证程序，以保证用户输入信息的完整性和正确性。

表单测试可以使用黑盒测试法中的等价类划分方法进行。例如某电子商务网站用户注册表单中需要填写 E-mail 地址，设计的表单测试用例如表 8-1 所示。

表 8-1　表单测试

类　　别	等　价　类	测试用例输入	预 期 输 出
有效等价类	64 个字符组成的字符串，由英文字母、数字、“ ”或“.”组成，含有“@”和“.”，“.”在“@”之后	Liming123@yahoo.com	无
无效等价类	超过 64 个字符	Lim..(超过 64 个字符)	邮件地址格式错误
	含有非法字符	Limin * 123@yahoo.com	邮件地址格式错误
	无“@”	Liming123yahoo.com	邮件地址格式错误
	无“.”	Liming123@yahoocom	邮件地址格式错误
	“.”在”@”之前	Liming123yahoo.com@	邮件地址格式错误

表单提交以后，需要验证服务器是否能正确保存这些数据。此时可以检查服务器端数据库内容来进行验证。

3. Cookies 测试

Cookies 是一种能够让网站服务器将少量数据储存到客户端的硬盘或内存，或是从客户端的硬盘读取数据的一种技术。Web 应用程序可以使用 Cookies 存储用户信息，并在页面之间传递用户信息。

如果 Web 应用系统使用了 Cookies，在测试时必须检查 Cookies 程序是否能正常运行。Cookies 测试内容包括：指定的信息是否正确写入 Cookies 文件，Web 应用程序从 Cookies 文件读取的信息是否正确，Cookies 文件是否按预定的时间进行保存，Web 应用程序对 Cookies 中信息加密和解密过程是否正确，页面刷新对 Cookies 有何影响等。

Cookies 测试过程主要采用黑盒测试法，测试时可以运行建立和访问 Cookies 的页面，然后使用 IECookiesView 等 Cookies 查看工具，检查 Cookies 是否建立，Cookies 内容与期望值是否一致。

4. 数据库测试

在 Web 应用程序中，大量的信息存放在数据库中。用户通过提交表单完成对数据库的插入、删除、修改、查询等工作。目前在 Web 应用程序中使用比较广泛的是关系型数据库，例如 SQL Server、Access 等，页面通过调用 SQL 语句对数据库中的信息进行处理。Web 应用程序中数据库测试目的是保持数据的完整性和程序并发执行时数据的正确性。

数据的完整性主要以数据库表为单位，检查数据库表以及表中各字段命名是否符合命名规范，表中字段是否完整，数据库表中的字段描述是否正确，包括字段的类型、长度，数据库表中的关系、索引、主键、约束是否正确。完整性测试可以先进行静态审查需求分析和概要设计文档中数据库分析、设计的说明，然后通过对数据库完成插入、删除、修改等操作进行测试。进行数据库测试时，往往需要向数据库中输入大量数据，然后对数据进行操作。手工输入大量数据费时费力，可以使用数据库测试工具自动生成大量模拟数据输入数据库，然后再进行测试。

由于 Web 应用程序为多用户并发执行程序，所以程序中通常需要采取措施处理并发问题。在进行数据库测试时，必须考虑并发操作与数据一致性问题，所以测试人员必须模拟两个以上用户同时读取或写入数据。

在并发性测试中，可以综合考虑手动和自动化测试技术。手动并发测试可以由两个测试人员同时对数据库进行读取和修改，检查执行结果是否正确。自动化测试可以使用 Web Application Stress、LoadRunner 等测试工具，模拟多个用户并发访问 Web 应用程序进行。

5. 设计语言测试

目前 Web 应用程序开发主要采用 HTML 语言和多种脚本语言。开发时采用的 HTML 版本与浏览器上的显示结果密切相关。如果用户使用较早版本的浏览器，可能无法解释和执行高版本 HTML 语言中的新增功能和标记。所以在功能测试中需要使用不同版本的浏览器对 Web 应用程序进行测试，检查不同的浏览器环境下 Web 应用程序的执行情况。另外，在分布式环境中，开发人员可能使用不同脚本语言如 Java、JavaScript、ActiveX、VBScript 或 Perl 等开发程序，所以在集成测试时应该进行验证。

基于 Web 的功能测试与传统的软件测试相比，对软件测试提出了新的要求。对于 Web 应用软件，除进行功能测试外，还需要进行安全性测试、可用性测试、压力测试等。成功的 Web 测试将使 Web 应用程序获得更高的质量。

8.3 功能测试的自动化

8.3.1 测试自动化框架

通常情况下，软件开发组织会使用自动化测试工具，使用录制回放方式来进行功能测试的自动化。但是录制回放方式并不能解决全部问题。所以自动化的功能测试需要有测试框架的支持。目前流行的测试自动化框架比较多，常见的有以下 5 种基本的软件测试框架。测试小组可以根据实际需要去考虑采用其中的一种测试框架，而不是仅依赖于一个简单的捕获工具。

1. 模块化测试框架

模块化测试脚本框架(test modularity framework)需要创建小而独立的可以描述的模块、片断以及待测应用程序的脚本。这些树状结构的小脚本组合起来，就能组成能用于特定的测试用例的脚本。

在 5 种框架中，这种应该是最容易掌握和使用的。在一个组件上建立一个抽象层，使其在余下的应用中隐藏起来，这是众所周知的编程技巧。它把应用同组件中的修改隔离开来，提供了程序设计的模块化特性。模块化测试脚本框架使用抽象或者封装的原理来提高自动化测试的可维护性和可升级性。

2. 测试库框架

测试库框架(test library architecture)与模块化测试脚本框架很类似，并且具有同样的优点。不同的是测试库框架把待测应用程序分解为过程和函数，而不是脚本。这个框架需要创建描述模块、片断以及待测应用程序的功能库文件，例如 SQA Basic libraries、APIs、DLL 等。

3. 关键字驱动的测试框架

这个框架需要开发数据表和关键字，这些数据表和关键字独立地执行它们的测试自动化工具，并可以用来“驱动”待测应用程序和数据的测试脚本代码。关键字驱动测试看上去与手工测试用例很类似。在一个关键字驱动测试中，把待测应用程序的功能和每个测试的执行步骤一起写到一个表中。这个测试框架可以通过很少的代码来产生大量的测试用例，同时这些代码还可以被复用。

4. 数据驱动测试框架

数据驱动测试是一个框架。在这个框架中，变量不仅被用来存放输入值，还被用来存放输出的验证值。整个程序中，测试脚本用来读取数值文件，记载测试状态和信息。在数据驱动测试中，数据文件中只包含测试数据。这个框架的目标是减少测试人员需要执行所有测试用例所需要的测试脚本总数。数据驱动需要很少的代码来产生大量的测试用例，这与关键字驱动极其类似。

5. 混合测试自动化(hybrid test automation)框架

最普遍的执行框架是上面介绍的所有技术的一个结合，取其长处，弥补其不足。这个混合测试框架是由大部分框架随着时间并经过若干项目演化而来的。

尤其是对于需要不断添加新需求和不断升级的软件系统，传统的自动化框架，如录制回放技术使得脚本几乎需要从头开发，而复杂的自动化框架，如关键字驱动则不容易为开发经验不足的测试工程师所理解，而导致难以根据新需求修改脚本。

8.3.2 SAFS 框架介绍

这里主要介绍比较先进的自动化测试框架模型(Software Automation Framework Support，SAFS)，它是由 SAS Institute 的 Carl Nagle 开发的，是一个关键字驱动的自动化测试框架，它的目标是试图建立一个与平台和执行工具无关的引擎。IBM 公司的 Rational Robot 工具就是利用这个模型开发了核心数据引擎，并实现了功能测试。

图 8-1 显示这个自动化测试模型的结构。这个模型主要由核心数据驱动引擎、成员函数、支持库和映射表组成。测试首先由初始脚本开始执行，这个脚本把高层测试表传递给高层驱动器，高层驱动器在处理这些表的过程中，遇到中层测试表就调用中层驱动器，中层驱动器处理中层表时也作类似的处理。当低层驱动器处理低层表的时候，它试着使

应用与测试保持同步。当低层驱动器遇到对某一个成员的低层关键字指令时,它判断这个成员的类型并调用相应的成员函数模块来处理这个指令操作。所有这些元素都要依靠映射表中的信息,它是自动化模型和被测应用之间的桥梁。

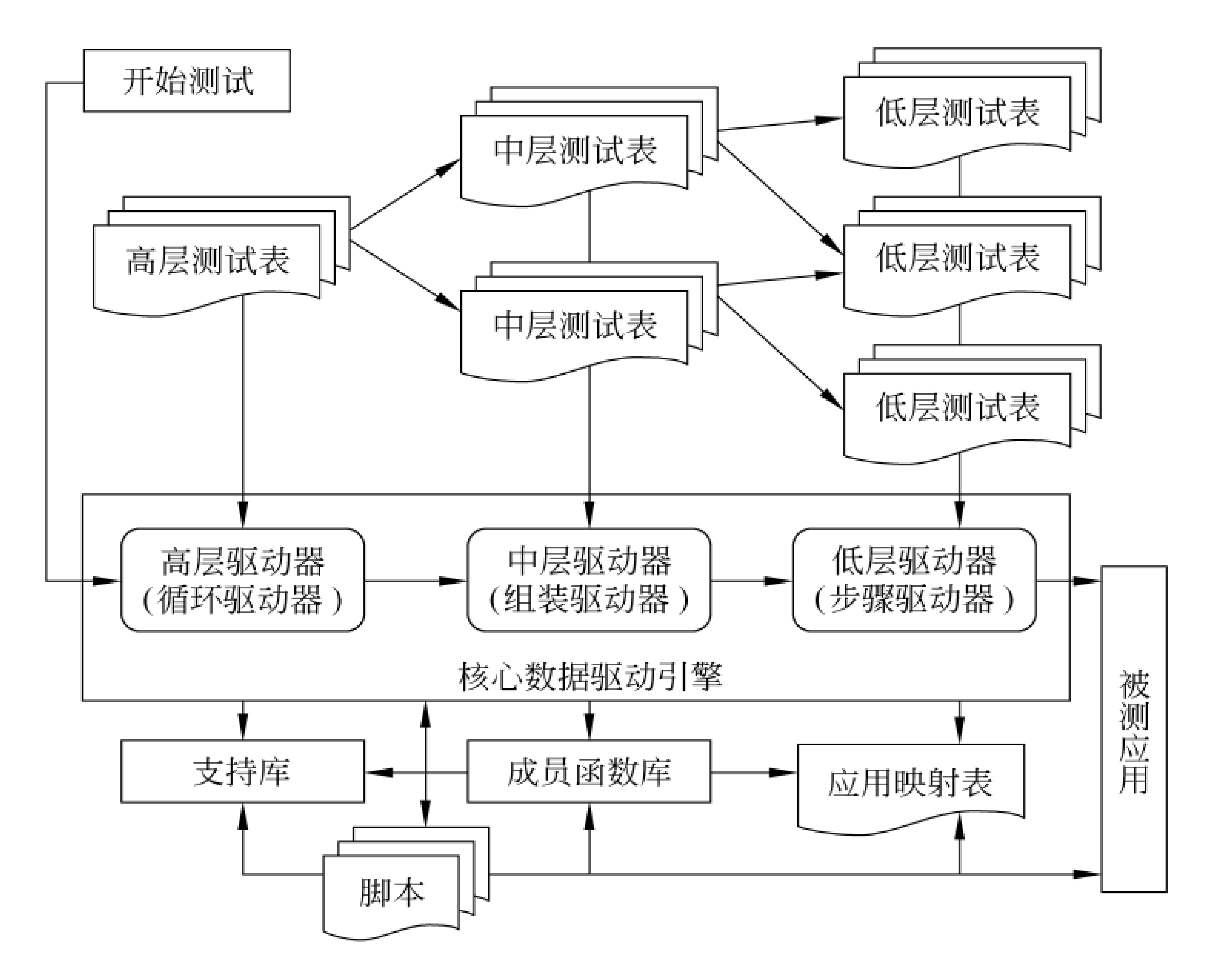


图 8-1 软件测试自动化模型的结构

1. 应用映射表(application map)

映射表是自动化模型中最关键的组件之一。在进行功能测试设计之前,测试人员首先对应用中的每一个对象定义一套命名规范,并利用映射表把这些名字和自动化工具识别的对象名联系起来,使工具能准确地定位和操纵对象,脚本只需要进行单点维护。在上面的例子中,如果按钮的名字或显示文字发生了变化,那么脚本中所有涉及这个按钮的地方都要进行修改。如果建立这样一个映射,用逻辑对象 SavePushButton 表示真实的确认保存的按钮对象,那么这个例子就可以写成 Click SavePushButton。当按钮的名字或显示文字改变时,只需要快速修改一下映射表中对应的识别方法就可以了,而不用修改脚本,如表 8-2 所示。

表 8-2 按钮对象的映射

命　　名	识 别 方 法
SavePushButton	Type=PushButton; Name=Yes; VisualText=Yes

2. 成员函数(component function)

成员函数是实现用户对界面对象操作指令的函数,一个成员对象的类型对应一个成员函数库。例如对于一个文本框对象,测试人员可能会对它执行多种操作:输入文本,验

证文本框的值,验证文本框的某些属性等,实现这些操作行为的函数就被放在文本框的成员函数库中。一般的测试工具都提供了这样的函数,而程序员可以在其中加入额外的代码来检测错误、纠正错误和帮助同步,这类代码是实现无人监守的自动化测试所必需的。

成员函数相当于在应用和自动化工具之间提供了一个隔离层,如果没有这个隔离层,自动化工具本身的改变或提高就会影响已有的脚本,但是有了成员函数,可以增加一段修补代码来适应这些变化,转移对测试的破坏。成员函数关键字和它们的参数构成自动化模型最底层的词库,了解低层词库和映射表,就可以建立在它们基础上的测试表。

3. 测试表和核心数据驱动引擎

测试表分低层、中层和高层。低层测试表指定了测试的每一步指令的细节,这些指令都是直接作用在界面对象上的,是无法再细分的指令。中层测试表把低层表组装起来执行更多有用的任务。同一个低层表可以用于多个中层表,所以应该开发尽可能少的低层表,然后把它们按照不同的目的组装起来,实现最大的重用性。同样地,高层测试表把中层表组装起来,形成一个测试循环,每个循环是一个完整的测试用例,可以定制不同类型和数量的测试。例如打开网页、登录、关闭网页这三个动作可以用三个低层表来表示,每个表定义了实现相应动作的具体步骤,所以低层表又叫做步骤表。低层表中使用了映射表中定义的对象名和由成员函数定义的低层关键字词库。表 8-3 是一个实现登录动作的低层表。而这个表示"登录"的低层表关键字,很可能会出现在"验证错误登录"、"验证正确登录"、"验证空白登录"等中层表中,这些中层表合起来构成了"验证权限"高层表。

表 8-3 "登录"步骤表

窗　口	成　员	动　作	参　数
LoginPage	UserIDField	InputText	MyUserID
LoginPage	PasswordField	InputText	MyPassword
LoginPage	SubmitButton	Click	

对应于以上这三个测试表,核心数据驱动引擎相应地分成了高层驱动器、中层驱动器和低层驱动器。高层驱动器读取高层表的每个记录,如果遇到中间表关键字,就把这个表传递给中层驱动器,依此类推,直至到达低层表,低层驱动器调用关键字词库中的低层指令所对应的成员函数来完成最后的执行。最后要说明的是这样一种层次结构并不是固定不变的,可以根据实际应用进行调整。

4. 支持库(support libraries)

支持库是一般辅助程序和工具,例如文件处理、字符串处理、缓冲处理、数据库访问、日志记录工具等,它们为自动化模型提供最基础的支持。

由以上可以看出,它使自动化测试与手工测试达到完美的结合。测试执行人员所要做的仅是输入在 Robot 中要执行的手工测试用例名称,以及设定执行次数等参数。最后的工作就是等待自动化测试执行完成后,获取到相应的执行结果进行分析。测试执行人员在整个过程中是不必知道自动化测试框架内部是如何运作的。

8.4 正则表达式

在编写处理字符串的程序或网页时，经常会有查找符合某些复杂规则的字符串的需要。正则表达式就是用于描述这些规则的工具。换句话说，正则表达式就是记录文本规则的代码。

很可能读者使用过 Windows/DOS 下用于文件查找的通配符（wildcard），也就是“*”和“?”。如果读者想查找某个目录下的所有的 Word 文档的话，读者会搜索*.doc。在这里，“*”会被解释成任意的字符串。与通配符类似，正则表达式也是用来进行文本匹配的工具，只不过比起通配符，它能更精确地描述读者的需求，但是代价变得更复杂，例如读者可以编写一个正则表达式，用来查找所有以 0 开头，后面跟着 2～3 个数字，然后是一个连字符“-”，最后是 7 或 8 位数字的字符串（如 010-12345678 或 0376-7654321）。

正则表达式是用于进行文本匹配的工具，所以这里多次提到了在字符串里搜索/查找，意思是在给定的字符串中，寻找与给定的正则表达式相匹配的部分。有可能字符串里有不止一个部分满足给定的正则表达式，这时每一个这样的部分被称为一个匹配。匹配一般有三种意思：

- 形容词性的，例如一个字符串匹配一个表达式；
- 动词性的，例如在字符串里匹配正则表达式；
- 名词性的，例如字符串中满足给定的正则表达式的一部分。

匹配要根据上下文环境来判断。

例 8.1 在一篇英文小说里查找 hi，该使用什么表达式？

解析：要在一篇英文小说里查找 hi，可以使用正则表达式 hi。这是最简单的正则表达式了，它可以精确匹配这样的字符串：由两个字符组成，前一个字符是 h，后一个是 i。通常处理正则表达式的工具会提供一个忽略大小写的选项，如果选中了这个选项，它可以匹配 hi、HI、Hi、hI 这四种情况中的任意一种。

很多单词里包含 hi 这两个连续的字符，例如 him、history、high 等。用 hi 来查找的话，这里的 hi 也会被找出来。如果要精确地查找 hi 这个单词的话，应该使用\bhi\b。

\b 是正则表达式规定的一个特殊代码（也叫元字符），代表着单词的开头或结尾，也就是单词的分界处。通常英文的单词是由空格或标点符号或换行来分隔的，但是\b 并不匹配这些单词分隔符中的任何一个，它只匹配一个位置。

假如要找的是 hi 后面不远处跟着一个 Lucy，应该用\bhi\b.*\bLucy\b。

这里，“.”是另一个元字符，匹配除了换行符以外的任意字符。“*”同样是元字符，不过它代表的不是字符，也不是位置，而是数量——它指定“*”前边的内容可以连续任意重复出现若干次，以使整个表达式得到匹配。因此，“.*”连在一起就意味着任意数量的不包含换行的字符。现在\bhi\b.*\bLucy\b 的意思就很明显了：先是一个单词 hi，然后是任意个任意字符（但不能是换行），最后是 Lucy 这个单词。

例 8.2 怎么查找跟 021-11223344 相似的电话号码？

解析：可使用 0\d\d-\d\d\d\d\d\d\d\d 匹配这样的字符串：以 0 开头，然后是两个数字，然后是一个连字符“-”，最后是 8 个数字，当然，这个例子只能匹配区号为 3 位的情形。

这里的\d 是一个新的元字符，匹配任意的数字(0，1，2，…)。“-”不是元字符，只匹配它本身——连字符号。

为了避免烦琐的重复，也可以这样写这个表达式：0\d{2}-\d{8}。这里\d 后面的{2}({8})的意思是前面\d 必须连续重复匹配 2 次(8 次)。

由以上例子可以看到，如果同时使用其他的一些元字符，就能构造出功能更强大的正则表达式。

8.4.1 测试正则表达式

正则表达式的语法很复杂，即使对经常使用它的人来说也是如此。由于难于读写，容易出错，所以很有必要创建一种工具来测试正则表达式。

测试员还可以通过在验证点中使用正则表达式，建立更加灵活的验证点，保证测试脚本的重用性，如图 8-2 所示。

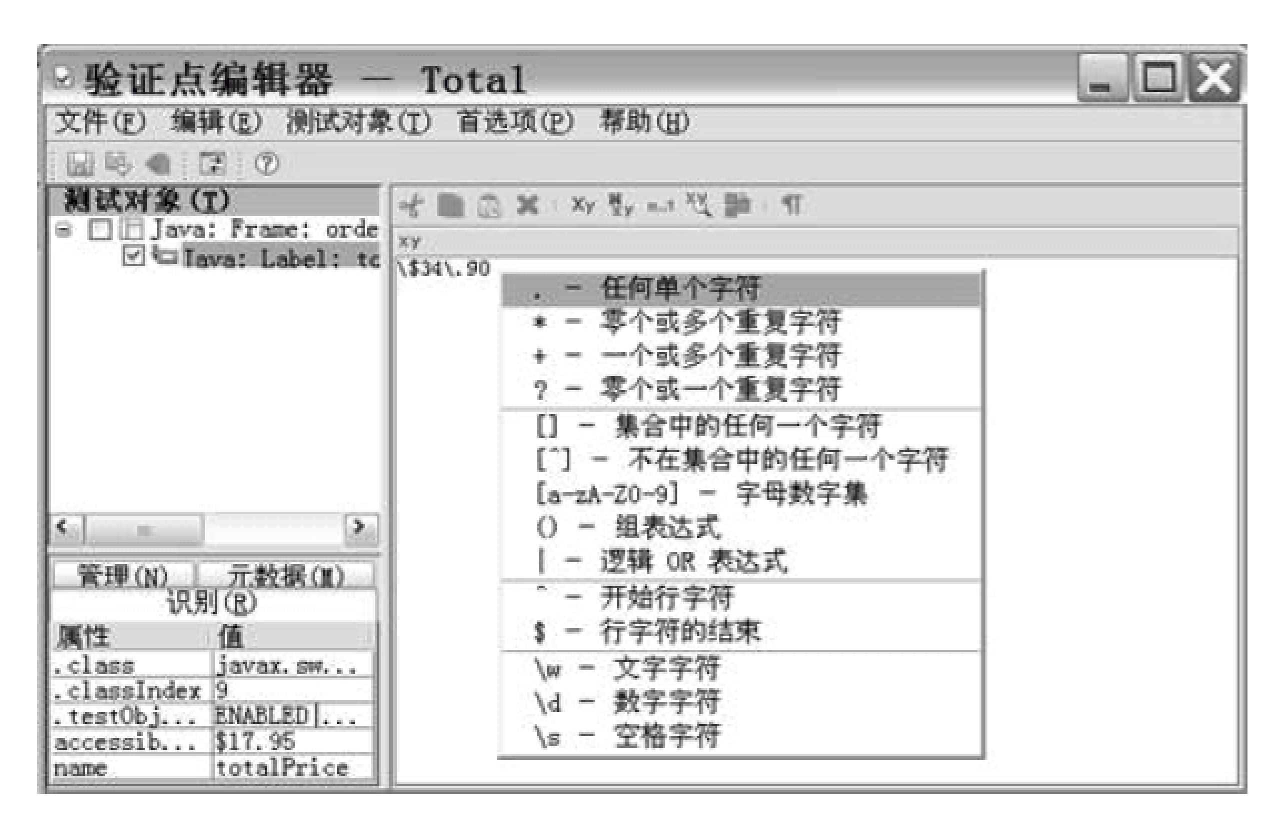


图 8-2 正则表达式在验证点中的应用

8.4.2 元字符

现在已经知道几个很有用的元字符了，如\b、.、*，还有\d。当然还有更多的元字符可用，例如\s 匹配任意的空白符号，包括空格、制表符、换行符、中文全角空格等。\w 匹配字母或数字或下划线或汉字等，如表 8-4 所示。

例 8.3 如何查找以 a 开头的单词?

解析：可使用\ba\w*\b 正则表达式来匹配以字母 a 开头的单词，先是某个单词开始处(\b)，然后是字母 a，然后是任意数量的字母或数字(\w*)，最后是单词结束处(\b)。

\d+匹配 1 个或更多连续的数字。这里的“+”是和“*”类似的元字符，不同的是“*”匹配重复任意次(可以是 0 次)，而“+”则匹配重复 1 次或更多次。

表 8-4　常用的元字符

代　码	说　　明	代　码	说　　明
.	匹配除换行符以外的任意字符	\b	匹配单词的开始或结束
\w	匹配字母或数字或下划线或汉字	^	匹配字符串的开始
\s	匹配任意的空白符号	$	匹配字符串的结束
\d	匹配数字		

\b\w{6}\b 匹配刚好 6 个字母/数字的单词。

元字符^和 $ 与\b 有点类似，都匹配一个位置。^匹配要用来查找的字符串的开头，$ 匹配结尾。这两个代码在验证输入的内容时非常有用，例如一个网站如果要求填写的 QQ 号必须为 5 位到 12 位数字时，可以使用：^\d{5,12}$。

这里的{5,12}和前面介绍过的{2}是类似的，只不过{2}匹配只能不多不少重复 2 次，{5,12}则是重复的次数不能少于 5 次，不能多于 12 次，否则都不匹配。

因为使用了^和 $，所以输入的整个字符串都要用来和\d{5,12}匹配，也就是说整个输入必须是 5～12 个数字，因此如果输入的 QQ 号能匹配这个正则表达式的话，那就符合要求了。

它与忽略大小写的选项类似，有些正则表达式处理工具还有一个处理多行的选项。如果选中了这个选项，^和 $ 的意义就变成了匹配行的开始处和结束处。

8.4.3　字符转义

如果想查找元字符本身的话，例如查找“.”或“*”，这时就必须使用\来取消这些字符的特殊意义。因此应该使用\. 和*。当然要查找\本身也得用\\。

例如：www\.unibetter\.com 匹配 www.unibetter.com，c:\\Windows 匹配 c:\Windows。

8.4.4　重复

前面已讲过*、+、{2}、{5,12}这几个匹配重复的方式。表 8-5 是正则表达式中所有的限定符(指定数量的代码)，例如*、{5,12}等。

表 8-5　常用的限定符

代码/语法	说　　明	代码/语法	说　　明
*	重复 0 次或更多次	{n}	重复 n 次
+	重复 1 次或更多次	{n,}	重复 n 次或更多次
?	重复 0 次或 1 次	{n,m}	重复 n 到 m 次

例 8.4　Windows\d+匹配 Windows 后面跟 1 个或更多数字。

例 8.5　13\d{9}匹配 13 后面跟 9 个数字(中国的手机号)。

例 8.6　^\w+匹配一行的第 1 个单词或整个字符串的第 1 个单词，具体匹配哪个得

看选项设置。

8.4.5 字符类

例 8.7 如何匹配任何一个英文元音字母?

解析：要想查找数字、字母或空白是很简单的，因为已经有了对应这些字符集合的元字符，但是如果想匹配没有预定义元字符的字符集合(例如元音字母 a,e,i,o,u)，这时只需要在中括号里列出它们就行，例如使用[aeiou]就匹配任何一个英文元音字母，[.?!]匹配标点符号(英文语句通常只以这三个标点结束)。

也可以轻松地指定一个字符范围，像[0-9]代表的含意与\d 就是完全一致的：一位数字，同理[a-z0-9A-Z_]也完全等同于\w(如果只考虑英文的话)。

例 8.8 如何匹配几种格式的电话号码，如(010)88886666,022-22334455,02912345678。

解析：可使用\(? 0\d{2}[) -]? \d{8}这个更复杂的表达式。这个表达式可以匹配几种格式的电话号码，如(010)88886666,022-22334455,02912345678。这个表达式首先是一个转义字符"\(",它能出现 0 次或 1 次"?",然后是一个 0，后面跟着 2 个数字(用\d{2}表示)，然后是")"或"-"或空格中的一个，它出现 1 次或不出现"?",最后是 8 个数字(用\d{8}表示)。当然这个表达式还有个问题，就是它也能匹配 010)12345678 或(022-87654321 这样的"不正确"的格式。

8.4.6 反义

有时需要查找不属于某个能简单定义的字符类的字符。例如想查找除数字外其他任意字符的情况，这时需要用到反义，如表 8-6 所示。

表 8-6 常用的反义代码

代码/语法	说　　明
\W	匹配任意不是字母、数字、下划线、汉字的字符
\S	匹配任意不是空白符的字符
\D	匹配任意非数字的字符
\B	匹配不是单词开头或结束的位置
[^x]	匹配除了 x 以外的任意字符
[^aeiou]	匹配除了 aeiou 这几个字母以外的任意字符

例如，\S＋匹配不包含空白符的字符串，＜a[^＞]＋＞匹配用尖括号括起来的以 a 开头的字符串。

8.4.7 替换

正则表达式里的替换指的是几种规则，如果满足其中任意一种规则都应该当成匹配，

具体方法是用“|”把不同的规则分隔开。

例 8.9 如何匹配 3 位或 4 位区号格式的电话号码？

解析：可使用 0\d{2}-\d{8}|0\d{3}-\d{7}这个表达式能匹配两种以连字号分隔的电话号码：一种是三位区号，8 位本地号(如 010-12345678)，一种是 4 位区号，7 位本地号(0376-2233445)。

\(0\d{2}\)[-]? \d{8}|0\d{2}[-]? \d{8}这个表达式匹配 3 位区号的电话号码，其中区号可以用小括号括起来，也可以不用，区号与本地号间可以用连字号或空格间隔，也可以没有间隔。可以试用替换“|”把这个表达式扩展成也支持 4 位区号的。

\d{5}-\d{4}|\d{5}这个表达式用于匹配美国的邮政编码。美国邮政编码的规则是 5 位数字，或者用连字号间隔的 9 位数字。之所以要给出这个例子是因为它能说明一个问题：使用替换时，顺序是很重要的。如果把它改成\d{5}|\d{5}-\d{4}的话，那么就只会匹配 5 位的邮政编码(以及 9 位邮政编码的前 5 位)。原因是匹配替换时，将会从左到右地测试每个分支条件，如果满足了某个分支的话，就不会去管其他的替换条件了。

8.4.8 分组

前面已经提到了怎么重复单个字符，即直接在字符后面加上限定符。但如果想要重复多个字符又该怎么办？这时可以用小括号来指定子表达式(也叫做分组)，然后就可以指定这个子表达式的重复次数了，也可以对子表达式进行其他一些操作(后面将介绍)。

(\d{1,3}\.){3}\d{1,3}是一个简单的 IP 地址匹配表达式。要理解这个表达式，请按下列顺序分析它：\d{1,3}匹配 1～3 位的数字，(\d{1,3}\.){3}匹配 3 位数字加上 1 个英文句号(这个整体也就是这个分组)重复 3 次，最后再加上 1 个 1～3 位的数字(\d{1,3})。

(\d{1,3}\.){3}\d{1,3}也将匹配 256.300.888.999 这种不可能存在的 IP 地址(IP 地址中每个数字都不能大于 255)。如果能使用算术比较的话或许能简单地解决这个问题，但是正则表达式中并不提供关于数学的任何功能，所以只能使用冗长的分组，选择字符类来描述一个正确的 IP 地址：((2[0-4]\d|25[0-5]|[01]? \d\d?)\.){3}(2[0-4]\d|25[0-5]|[01]? \d\d?)。

理解这个表达式的关键是理解 2[0-4]\d|25[0-5]|[01]? \d\d?。

8.4.9 后向引用

使用小括号指定一个子表达式后，匹配这个子表达式的文本(也就是此分组捕获的内容)可以在表达式或其他程序中作进一步的处理。默认情况下，每个分组会自动拥有一个组号，规则是：从左向右，以分组的左括号为标志，第一个出现的分组的组号为 1，第二个为 2，依此类推。后向引用用于重复搜索前面某个分组匹配的文本。例如，\1 代表分组 1 匹配的文本。

例 8.10 如何匹配重复的单词？例如 go go，kitty kitty。

解析：\b(\w+)\b\s+\1\b 可以用来匹配重复的单词。首先是一个单词，也就是单

词开始处和结束处之间的多于一个的字母或数字(\b(\w＋)\b)，然后是 1 个或几个空白符(\s＋)，最后是前面匹配的那个单词(\1)。

也可以指定子表达式的组名。要指定一个子表达式的组名，请使用这样的语法：(?＜Word＞\w＋)。或者把尖括号换成“'”也行：(? 'Word'\w＋)。这样就把\w＋的组名指定为 Word 了。要反向引用这个分组捕获的内容，可以使用\k＜Word＞，所以例 8.9 也可以写成这样：\b(? ＜Word＞\w＋)\b\s＋\k＜Word＞\b。

使用小括号的时候，还有很多特定用途的语法。表 8-7 列出了最常用的一些。

表 8-7　分组语法

捕获	(exp)	匹配 exp，并捕获文本到自动命名的组里
	(? ＜name＞exp)	匹配 exp，并捕获文本到名称为 name 的组里，也可以写成(? 'name'exp)
	(?:exp)	匹配 exp，不捕获匹配的文本，也不给此分组分配组号
零宽断言	(?＝exp)	匹配 exp 前面的位置
	(?＜＝exp)	匹配 exp 后面的位置
	(?! exp)	匹配后面跟的不是 exp 的位置
	(?＜! exp)	匹配前面不是 exp 的位置
注释	(? #comment)	这种类型的组不对正则表达式的处理产生任何影响，用于提供注释让人阅读

在捕获的三种语法中，已经讨论了前两种语法。第三种语法(?:exp)不会改变正则表达式的处理方式，只是这样的组匹配的内容不会像前两种那样被捕获到某个组里面。

8.4.10　零宽断言

四个断言可用于查找在某些内容(但并不包括这些内容)之前或之后的东西，也就是说它们像\b、^、$ 那样用于指定一个位置，这个位置应该满足一定的条件(断言)，因此它们也被称为零宽断言。

例 8.11　如何匹配以 ing 结尾的单词的前面部分？如查找 I'm singing while you're dancing. 时，它会匹配 sing 和 danc。

解析：可使用\b\w＋(?＝ing\b)表达式匹配以 ing 结尾的单词的前面部分(除了 ing 以外的部分)，(?＝exp)也叫零宽度正预测先行断言，它断言自身出现的位置的后面能匹配表达式 exp。

例 8.12　如何匹配以 re 开头的单词的后半部分(除了 re 以外的部分)，例如在查找 reading a book 时，它匹配 ading。

解析：可使用(? ＜＝\bre)\w＋\b 表达式匹配以 re 开头的单词的后半部分(除了 re 以外的部分)。(? ＜＝exp)也叫零宽度正回顾后发断言，它断言自身出现的位置的前面能匹配表达式 exp。

假如想要给一个很长的数字中每三位间加一个逗号(从右边加起)，可以这样查找需要在前面和里面添加逗号的部分：((? ＜＝\d)\d{3})＊\b，用它对 1234567890 进行查

找时结果是 234567890。

8.4.11 负向零宽断言

前面提到过怎么查找不是某个字符或不在某个字符类里的字符的方法(反义)。但是如果只是想要确保某个字符没有出现,但并不想去匹配它时怎么办?

例 8.13 如果想查找这样的单词,它里面出现了字母 q,但是 q 后面跟的不是字母 u,该怎么办?

解析:可以尝试这样:\b\w*q[^u]\w*\b 匹配包含后面不是字母 u 的字母 q 的单词。但是如果多做测试,可以发现,如果 q 出现在单词的结尾的话,像 Iraq,Benq,这个表达式就会出错。这是因为[^u]总要匹配一个字符,所以如果 q 是单词的最后一个字符的话,后面的[^u]将会匹配 q 后面的单词分隔符(可能是空格或句号或其他),后面的\w*\b 将会匹配下一个单词,于是\b\w*q[^u]\w*\b 就能匹配整个 Iraq fighting。负向零宽断言能解决这样的问题,因为它只匹配一个位置,并不消费任何字符。现在我们可以这样来解决这个问题:\b\w*q(?! u)\w*\b。

零宽度负预测先行断言(?! exp),断言此位置的后面不能匹配表达式 exp。例如\d{3}(?! \d)匹配三位数字,而且这三位数字的后面不能是数字,\b((?! abc)\w)+\b 匹配不包含连续字符串 abc 的单词。

同理,可以用(? <! exp),零宽度正断言来断言此位置的前面不能匹配表达式 exp:(? <! [a-z])\d{7}匹配前面不是小写字母的七位数字。

例 8.14 如何匹配不包含属性的简单 HTML 标签内里的内容?

解析:可使用(? <=<(\w+)>).*(? =<\/\1>)匹配不包含属性的简单 HTML 标签内里的内容。(<? (\w+)>)指定了这样的前缀:被尖括号括起来的单词(例如可能是),然后是.*(任意的字符串),最后是一个后缀(? =<\/\1>)。注意后缀里的\/,它用到了前面提过的字符转义;\1 则是一个反向引用,引用的正是捕获的第一组,前面的(\w+)匹配的内容,这样如果前缀实际上是的话,后缀就是了。整个表达式匹配的是和之间的内容(不包括前缀和后缀本身)。

8.4.12 注释

小括号的另一种用途是通过语法(? #comment)来包含注释。例如:2[0-4]\d(? #200-249)|25[0-5](? #250-255)|[01]? \d\d? (? #0-199)。

要包含注释的话,最好是启用“忽略模式里的空白符”选项,这样在编写表达式时能任意地添加空格、Tab、换行符,而实际使用时这些都将被忽略。启用这个选项后,在#后面到这一行结束的所有文本都将被当成注释忽略掉。

例如,可以将前面的一个表达式写成这样:

```
(?<=          #断言要匹配的文本的前缀
<(\w+)>       #查找尖括号括起来的字母或数字(即 HTML/XML 标签)
```

```
)                   #前缀结束
.*                  #  匹配任意文本
(?=                 #断言要匹配的文本的后缀
<\/\1>              #查找尖括号括起来的内容：前面是一个"/",后面是先前捕获的标签
)                   #后缀结束
```

8.4.13 贪婪与懒惰

当正则表达式中包含能接受重复的限定符时，通常的行为是（在使整个表达式能得到匹配的前提下）匹配尽可能多的字符。考虑这个表达式 a.*b，它将会匹配最长的以 a 开始，以 b 结束的字符串。如果用它来搜索 aabab 的话，它会匹配整个字符串 aabab。这被称为贪婪匹配。

相对贪婪匹配，有时更需要懒惰匹配，也就是匹配尽可能少的字符。前面给出的限定符都可以被转化为懒惰匹配模式，只要在它后面加上一个问号?。这样.*? 就意味着重复匹配任意数量，但是在能使整个匹配成功的前提下使用最少的重复。

例 8.15 如何匹配最短的，且以 a 开始、以 b 结束的字符串？

解析：a.*?b 匹配最短的，且以 a 开始、以 b 结束的字符串。如果把它应用于 aabab 的话，它会匹配 aab 和 ab。为什么第一个匹配是 aab 而不是 ab？简单地说，因为正则表达式有另一条规则，比懒惰/贪婪规则的优先级更高，最先开始的匹配拥有最高的优先权。懒惰限定符如表 8-8 所示。

表 8-8 懒惰限定符

*?	重复任意次，但尽可能少重复
+?	重复 1 次或更多次，但尽可能少重复
??	重复 0 次或 1 次，但尽可能少重复
{n,m}?	重复 n 到 m 次，但尽可能少重复
{n,}?	重复 n 次以上，但尽可能少重复

8.5 Robot 测试实践

这里以一款优秀的功能测试工具 IBM Rational Robot 为例进行介绍，它被评为 2002 年 Yphise 奖最佳功能测试工具。它可以在测试人员学习脚本技术之前，帮助其进行自动化测试，也可以帮助经验丰富的测试人员来修改测试脚本，改进测试深度。其功能测试架构如图 8-3 所示。

业务测试人员类似于当前软件开发组织中使用手工执行测试的测试人员。可以看到，在解决方案中，除传统的业务测试人员外，增加了技术测试人员角色。技术测试人员偏重于自动化测试相关技术，实际上并不直接执行测试。

基于 Robot 的功能测试构架的核心是使用前面介绍的 SAFS 框架，此框架以关键字

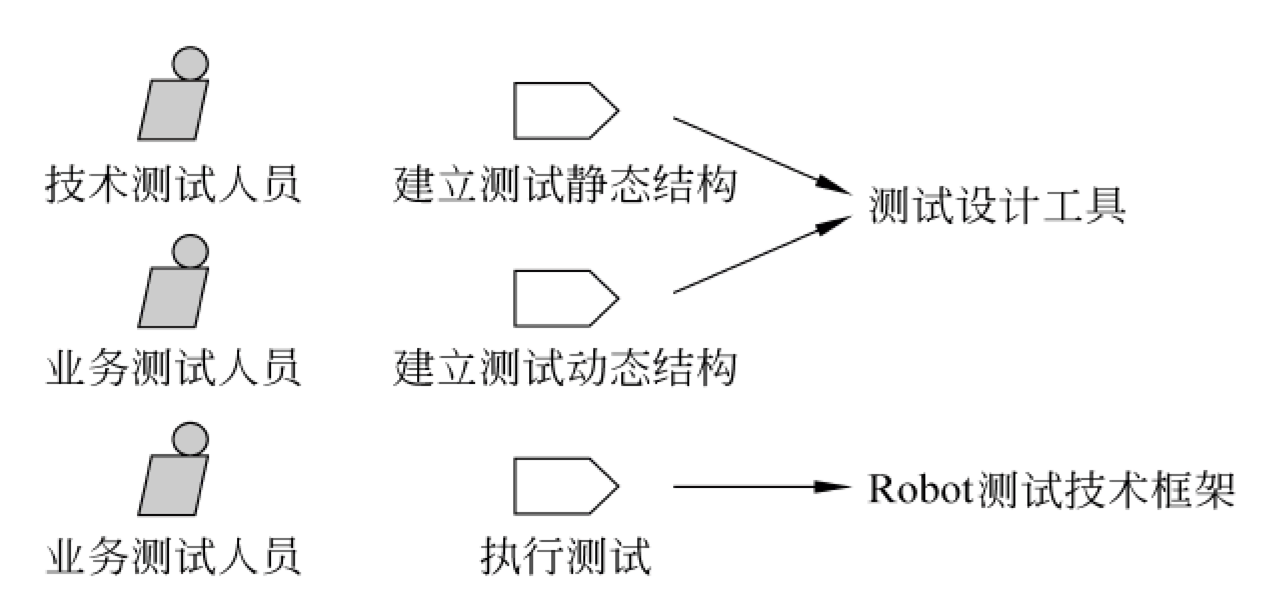


图 8-3　基于 Robot 的功能测试构架

驱动为指导思想，读入动态结构，解释并执行动态结构中的每一项，是自动化测试的引擎。同时，为了提高 SAFS 框架的易用性，Robot 功能测试构架中还包括测试设计工具，它是使用其他编程语言如 Java、Delphi 等开发的应用程序。在测试设计工具中，测试技术人员首先建立和待测试应用一一对应的静态结构，此静态结构以页面为单位，随后业务测试人员从静态结构中选择不同的页面，组成测试动态结构，即测试用例，随后此动态结构被 SAFS 框架读入并解释执行。

本节将分别展示对象识别、验证点、数据池、分支执行、数据关联、日志记录、脚本在 SAFS 框架中的实现方式。

8.5.1　关键字驱动实践

SAFS 框架是基于关键字驱动测试思想。关键字驱动测试就是预先在表中清楚定义的代表每一步执行操作的关键字，然后由脚本读入表中的每一行，根据关键字来执行对应的动作。下面以图 8-4 所示的 CQ Web 登录界面为例。



图 8-4　ClearQuest Web 登录界面

当要自动执行“登录”按钮时，可以如下来定义。

登录后在 Robot 的脚本中，打开表，读入此行并执行。这样的话，Robot 便去单击界面上的“登录”按钮了。Robot 可以监测到测试人员与应用程序之间的所有交互行为，并可以产生相应的测试脚本。

其脚本如下：

```
'打开文件
Dim sData() as string
InFileName=getExcelFileName
ReadExcelData InFileName, sData()

'解释并执行
Select Case (sKeyWord)
Case "登录"
    Window SetContext, "currentwindow", ""
    PushButton Click, "Text=登录", ""
```

这个简单的脚本一旦录制完成，就可以直接执行它。当然大多数情况下，测试脚本一般需要改进与增强。改进和增强测试脚本的工作非常简单，就像在程序代码中添加几行代码以处理一些条件逻辑那样简单。这对于有一点开发语言基础的人来说，也是很容易的工作。

例 8.16 测试在给定的环境中计算机屏幕上是否弹出了一个窗口。

在这个例子中，只需要在测试脚本的代码中添加简单的类型声明，以处理窗口是否出现。

细心的读者还可以观察到，这个脚本就是 Visual Basic 语言，也就是测试脚本编程语言 SQA Basic，它是从 Visual Basic 语言中演化而来的，同时对语法进行了扩展，添加了一些测试专用的命令。这些新的命令扩展了 Robot 对所有 GUI 对象的编程访问能力，同时也使通常的编程任务像创建一个数据驱动的测试一样更加简单。通过使用这种语言，即便是很少编程经验的测试人员，也能够很容易理解代码的含义。对于那些有丰富编程经验的人来说，他们将会发现 SQA 可以非常灵活地进行一些高级的编程，例如利用 COM 对象或者访问 Windows 的编程接口。

一旦完成了录制和改进测试脚本，就可以开始执行脚本来完成测试了。在执行或回放时，Robot 重复所有的用户交互，计算当前的应用程序结果与验证基线的任何差异，并将结果记录在测试日志中。在所有的测试脚本被执行完后，QA 小组检查测试日志，评估他们应用程序的健康性。

成功的脚本执行的关键在于拥有多执行点的能力。有时测试人员可能希望只是执行单个或少量的脚本，其他时候希望执行所有的测试用例。这两种情况需要不同的考虑。

Robot 可以以如下方式执行测试脚本(见图 8-5)：

- 从 Robot 图形界面中执行脚本；
- 从 Robot 命令行中执行脚本；
- 从 TestManager 中执行脚本(具有远程执行脚本的能力)。

8.5.2 Robot 的对象识别

根据 IBM Rational Robot 识别对象并执行操作的要求，如果要让 Robot 找到界面上的对象并执行相关动作，需要给 Robot 指定每个对象的对象类型、对象标志、执行动作和



图 8-5　Robot 执行测试的方式

数据，如图 8-6 所示。

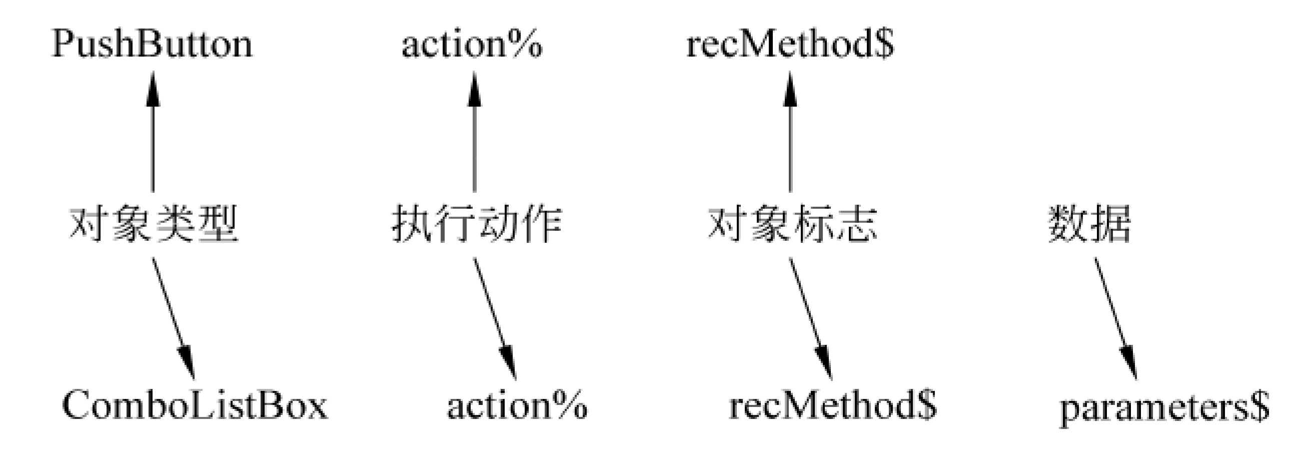


图 8-6　为 Robot 指定每个对象的对象类型、对象标志、执行动作和数据

以按钮来讲，如果要让 Robot 自动单击某个按钮，那么，首先要告诉 Robot 需要在 Button 这种类型的对象上进行操作；其次要告诉 Robot，在此类型的对象上要执行什么操作，例如单击；再次要告诉 Robot 要单击那个具体的按钮上，例如要单击“登录”按钮。

对象识别表如表 8-9 所示。

8-9　对象识别表 1

动作类型	对象类型	对象标志	执行动作	数据
G	Button	确定	Click	
G	EditBox	姓名	Click	Jack
G	ComboBox	角色列表	Click	系统管理员
G	RadioButton	区域	Click	

表 8-9 中的脚本如下：

```
'打开文件
Dim sData() as string
InFileName=getExcelFileName
ReadExcelData InFileName, sData()

'对文件中每一行
Select Case (sObjType)
Case "Button"
    ProcessButton(sObjAction, sObjData, sData)
Case "EditBox"
```

```
        ProcessEditBox(sObjAction, sObjData, sData)
    Case "ComboBox"
        ProcessComboBox(sObjAction, sObjData, sData)
    Case "RadioButton"
        ProcessRadioButton(sObjAction, sObjData, sData)

'对按钮执行的动作
Select Case(sObjAction)
    Case "Click"
        Window SetContext, "currentwindow", ""
        PushButton Click,  "Text="&sObjData, ""

'对文本框执行的动作
Select Case(sObjAction)
    Case "Click"
        EditBox Click, "Name="&sObjData, ""
        InputKeys "^+{HOME}{DELETE}"
        InputKeys sData

'对组合框执行的动作
Select Case(sObjAction)
    Case "Click"
        ComboBox Click, "Name="&sObjData, ""
        ComboListBox Click, "Name="&sObjData, "Text="&sData

'对单选按钮执行的动作
Select Case(sObjAction)
    Case "Click"
        RadioButton Click, "Name="&sObjData
```

需要强调的是，以按钮为例，虽然在表 8-9 中需要为界面上每一个具体的按钮定义一行，但是在测试技术框架中，所有按钮处理的代码都是一样的。

8.5.3 验证点

没有验证点的自动化测试就不能称之为测试。从这句话中就可以看到验证点在自动化测试中的重要性。

什么是验证点？以 HTTP 协议的脚本来说明，发送一个请求到服务器后，如果服务器不返回类似 http500 的错误，那么工具会认为这次请求是成功的，至于是否返回了用户期望的结果，工具自己是无法判断的。而验证点就提供这样一个功能，来对比实际结果和预期的结果之间的差异。在 Robot 中最常用的验证点是对象属性和对象的数据验证。这些验证点被用于捕获对象的状态和对象测试期间的数据。在 Robot 中创建验证点与选择想得到的验证点以及识别想要被测试的对象一样简单。

因此，验证点是脚本中非常重要的组成部分，它完成对被测试程序生成的实际数据和期望数据的比较，并将比较结果写入日志。一般情况下，测试的结果是通过对验证点的执行而得到的。

但是很多情况下想要的验证点可能并不是可以看到的控件。就像图 8-7 中显示的，测试者看到的是浏览器中各个元素的结果值，这些结果值 Robot 也可以看到，但测试者却看不到网页上对象的属性，例如网页的 Cookie 属性，这些对象属性都可以被 Robot 看见。



图 8-7　Robot 的测试验证点

Robot 提供了 13 种类型的验证点，这些验证点可以分类，如表 8-10 所示。

表 8-10　数据验证点

验证点类型	期望数据	实际值
静态验证点	记录在脚本中	在脚本回放中捕获
手动验证点（一个数据源）	在第一次脚本运行中得到的值	手动插入验证点的数据
手动验证点（两个数据源）	手动插入验证点的数据	手动插入验证点的数据
动态验证点（不依靠测试对象）	在脚本的首次回放中设置的验证点的值	在脚本后续运行中捕获的值
动态验证点（依靠提供的测试对象）	在脚本的首次回放中设置针对被提供测试对象的验证点的值	在脚本后续运行中被提供测试对象捕获的值

这里就验证点进行说明：

① 静态验证点（Static Verification Point）：是在录制（Record）脚本的过程中通过向导插入的验证点，它在脚本回放（Playback）的过程中自动被验证。

② 手动验证点（Manual Verification Point）：如果验证点所要验证的内容是由脚本开发人员在脚本中所提供的，则需要建立手动验证点对其进行验证。例如待验证数据来自外部数据源的情况，脚本开发人员需将数据读取后，以参数的形式显示传给验证点。

③ 动态验证点（Dynamic Verification Point）：是在脚本首次回放时建立的。验证点一旦建立，其行为就和静态验证点相同了。

如果以录制/回放模式使用 Robot 进行图形界面（GUI）的自动化回归测试，较常用的是静态验证点。而由于 Robot 的关键字驱动测试特性以及与其他 RUP 工具的良好集成，使之也是非图形化界面的功能测试的首选工具之一。在这些测试用例中，存在着大量

的用户自定义类型对象，这些被测试对象，并不能在录制过程中被插入对象映射表（ObjectMap）中，也就是不能使用静态验证点来进行验证，这就需要使用手动验证点来比较它们。

一旦验证点被捕获了，信息就会被存储在测试数据区域。在执行回放时，测试捕获的数据将与测试数据区域中的数据基线进行比较。如果比较结果有任何的不同，它们将被标记为"失败"，并被记录在测试日志中。Robot 还具有对整个网站的断裂链接（表格背景图片是网页不可缺少的部分，如果对表格背景图片进行移动或改名等操作时，没有提示更新链接，就会出现断裂链接）进行检查的能力，这也是通过设置验证点实现的。

对于验证点来讲，因为不同的测试、不同的应用验证点都不相同，所以 SAFS 框架仅提供了扩展的机制，不同的验证点可以通过扩展机制加入到测试技术框架中。

加入验证点之后，表 8-9 的定义改为表 8-11。

表 8-11　对象识别表 2

动作类型	对象类型	对象标志	执行动作	数　据
G	Button	确定	Click	
G	HTMLLink	链接	Click	
G	ComboBox	角色列表	Click	系统管理员
G	RadioButton	区域	Click	
V	VP	VP_SUM	VP_SUM	24

表 8-11 中最后一行是加入的验证点。所有的验证点其对象类型均为 VP，不同的验证点有不同的对象标志，表 8-11 中的验证点是 VP_SUM，验证点的基线数据为 24。

```
'对文件中每一行
Select Case (sObjType)
Case …
    Process…
Case "VP"
    ProcessVP(sObjAction, sData)

'对验证点执行的动作
g_VP_SUM_Baseline=sData
CallScript sObjAction

'验证点脚本的处理
sData=g_VP_SUM_Baseline
SQAGetProperty "", "", sActual
if sData=sActual then
    …
else
    …
end if
```

将验证点的基线数据放入全局变量 g_VP_SUM_Baseline 中，然后使用 CallScript 函数来调用验证点的脚本。对每一个验证点单独地创建一个脚本文件，脚本文件的名字和验证点的标志相同，都是 VP_SUM。虽然各个验证点脚本的内容都不相同，但一般的步骤是首先从全局变量 g_VP_SUM_Baseline 中取出基线数据，然后使用 SQAGetProperty 函数从界面上取实际的数据，再比较实际数据和基线数据。

8.5.4 数据池

数据池(Datapool)是一个测试数据集，它为脚本回放期间提供数据值给脚本变量。

数据池可以自动在大数据量的情况下(潜在的包含数个虚拟测试人执行上千条事务)提取测试数据给虚拟测试人。

数据池的作用有：

① 实现每个虚拟测试人能在脚本运行时发送实际数据(独一的数据)给服务器。

② 实现单一的虚拟测试人多次执行相同的事务，能在每次执行事务发送实际数据给服务器。如果在回放脚本期间不用数据源，每个虚拟测试人会发送相同的数据给服务器(此数据是记录脚本捕获下的数据)。

例如：假设在记录 VU 脚本时发命令数 53328 给数据库服务器，若有 100 个虚拟测试人在运行这个脚本，则命令数 53328 会给服务器发送 100 次。如果运用 Datapool，每个虚拟测试人会发送不同命令数给服务器。

数据池在自动化测试中使用率很高，通过使用数据池，可以通过简单的脚本完成大量数据的测试，缩短测试时间，提高测试效率和测试质量。数据池的建立有两种方式，录制 GUI 脚本时在测试工具中建立，并设置数据域的字段、类型，默认可以随机生成 100 组数据，但也可使用户自定义输入测试数据。

Robot 往往需要使用不同的数据来运行同一个测试，因此使用了数据池技术。在 Robot 里，数据池的增加比较简单，就是往表中增加表示数据的列，每一列代表一次测试执行所需要的数据，如表 8-12 所示。

表 8-12　数据池表 a

动作类型	对象类型	对象标志	执行动作	数据 1	数据 2
G	Button	确定	Click		
G	HTMLLink	链接	Click		
G	ComboBox	角色列表	Click	系统管理员	普通管理员
G	RadioButton	区域	Click		
V	VP	VP_SUM	VP_SUM	24	24

从表 8-12 中看到，“数据 1”这一列代表一次测试的执行所需要的数据，“数据 2”代表另外一次测试的执行所需要的数据。

在 SAFS 框架中，加入循环，按照数据列的数量来进行循环，每一个循环均从第一行执行到最后一行。

8.5.5 执行分支

执行分支在测试中，往往是同一个业务或功能，但是因为输入的数据、选择的条件不同，而具有不同的执行流程。执行分支的处理比较简单，就是在相应的数据列的位置上，填写代表忽略的特殊标志，例如 IGNORE，当测试执行到此动作时，判断其数据是否是 IGNORE，如果是，就不执行此动作而到下一个动作，如表 8-13 所示。

表 8-13 数据池表 b

动作类型	对象类型	对象标志	执行动作	数据 1	数据 2
G	Button	确定	Click		
G	HTMLLink	链接	Click		
G	ComboBox	角色列表	Click	系统管理员	普通管理员
G	RadioButton	区域	Click		
V	VP	VP_SUM	VP_SUM	24	IGNORE

从表 8-13 中看到，第一次执行会执行 VP_SUM 验证点，但是第二次执行，因为验证点相应的数据是 IGNORE，所以就不会执行 VP_SUM 验证点。

在 SAFS 框架中，在每次执行动作时，先判断其数据是不是 IGNORE 即可。

8.5.6 数据关联

在测试中，需要处理数据关联这种情况。数据关联是指前一个动作执行完成后，应用产生新的数据，在随后的动作中需要用到。因为这些数据是在执行的过程中由程序产生的，所以没有办法预先在表中准备，如表 8-14 所示。

表 8-14 数据池表 c

动作类型	对象类型	对象标志	执行动作	数据 1	数据 2
G	Button	确定	Click		
G	HTMLLink	链接	Click		
G	ComboBox	角色列表	Click	系统管理员	普通管理员
G	RadioButton	区域	Click		
V	VP	VP_SUM	VP_SUM	24	IGNORE
G	DC			DC_GETID	DC_GETID
G	EditBox	交易号	Click	DC_GETID	DC_GETID

从表 8-14 中看到，首先使用 DC_GETID 来将要关联的数据取出来，然后在需要使用此数据的地方，再使用 DC_SETID 赋值回去。

在 SAFS 框架中，取数据的处理如下：

```
'对文件中每一行
Select Case (sObjType)
```

```
Case …
    Process…
Case "DC"
    ProcessDC(sData)

'对数据关联执行的动作
CallScript sData

'数据关联中,获取数据脚本的处理
SQAGetProperty "","", g_DC_ID
```

对每一个数据关联,取数据单独地创建一个脚本文件,脚本文件的名字和数据关联的名字相同,例如说都叫 DC_GETID。虽然数据关联取数据脚本的内容各不相同,但是一般的步骤是使用 SQAGetProperty 函数从界面上取得数据,放入全局变量 g_DC_ID 中。

在 SAFS 框架中,赋值回去的处理如下:

```
'对文件中每一行
Select Case (sObjType)
Case …
    Process…
Case "EditBox"
    ProcessEditBox(sObjAction, sObjData, sData)

'对文本框执行的动作
Select Case(sObjAction)
    Case "Click"
        EditBox Click, "Name=" & sObjData, ""
        InputKeys "^+{HOME}{DELETE}"
        InputKeys g_DC_ID
```

即从全局变量 g_DC_ID 中取出数据,再输入到文本框中。

8.5.7 与 TestManager 的集成

单一或少量的脚本能从 Robot 图形界面中或从命令行中被执行。更加复杂的大量的测试脚本能够在 IBM Rational TestManager 工具中被创建和执行。

当从 TestManager 中执行测试时,可以在远程的机器上执行测试。通过在远程的机器上安装"测试代理",TestManager 可以与远程的机器进行通信,并计划在远程机器上进行测试脚本的执行,而这个远程机器可能是在隔壁房间或其他任何地方。

Robot 通过与 Rational TestManager 的集成(见图 8-8)可以实现:

- TestManager 可以协调测试执行的时间安排和测试脚本的依赖关系;
- 以中心控制的方式计划在多台远程的机器上执行测试;
- TestManager 可以对测试进行配置(如被制定到 Windows XP 平台上的测试只

能在 Windows XP 平台上执行)。

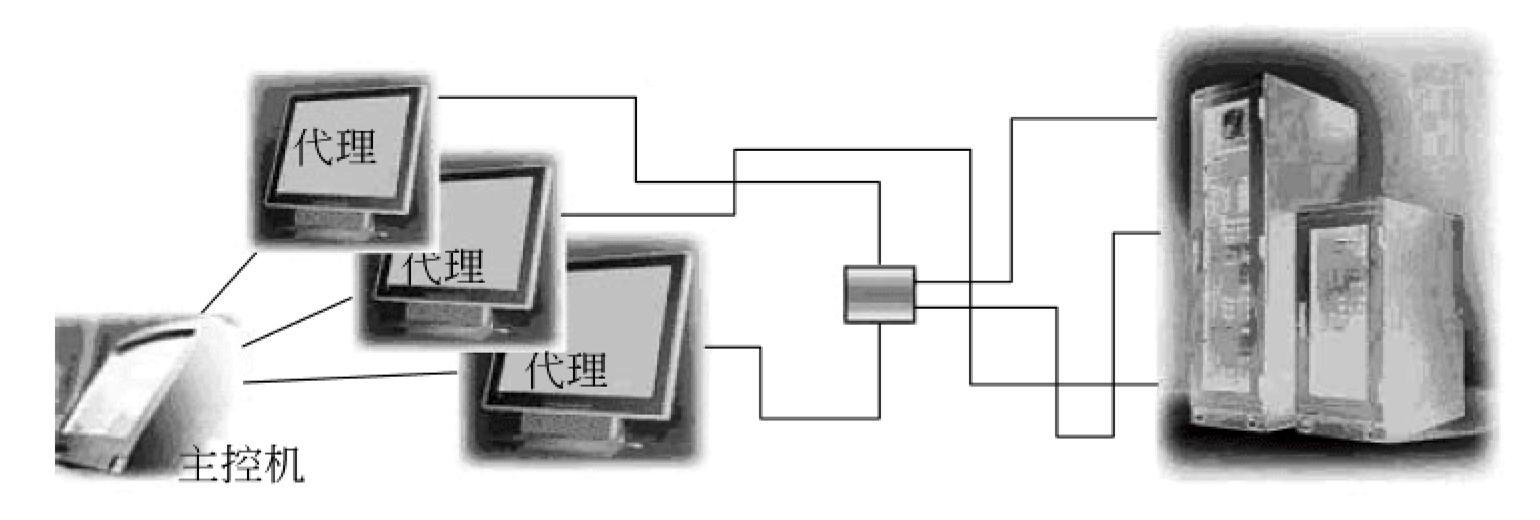


图 8-8　与 TM 集成进行测试

TestManager 提供了创建复杂的测试执行组合。TestManager 可以协调测试执行的时间安排和测试脚本的依赖关系。当回归测试的工作量不断增加时,这种能力是绝对必要的。

当从 TestManager 执行测试脚本时,TestManager 是“可配置的”,这就意味着当它计划在远程机器上执行一个测试脚本时,它对远程机器是可配置的(操作系统、处理器和其他任何条件)并针对配置来执行测试脚本。因为一个测试脚本需要对不同的操作系统有一些稍微不同的版本,例如 Windows 98 和 Windows XP。TestManager 将仅对被给定的测试代理配置发送正确的测试脚本。

8.5.8　其他处理

其他处理包括日志记录、调用其他脚本以及脚本结束,如表 8-15 所示。

表 8-15　数据池表 d

动作类型	对象类型	对象标志	执行动作	数据 1	数据 2
G	Button	确定	Click		
G	HTMLLink	链接	Click		
G	ComboBox	角色列表	Click	系统管理员	普通管理员
G	RadioButton	区域	Click		
V	VP	VP_SUM	VP_SUM	24	IGNORE
G	DC			DC_GETID	DC_GETID
G	EditBox	交易号	Click	DC_GETID	DC_GETID
L				输入交易号	输入交易号
S				Order	Order
X					

可以看到,在动作类型这一列,使用 L 代表记录日志,日志的内容存放在这一行数据列中,例如表 8-15 中的“输入交易号”;使用标志 S 代表调用其他脚本,要调用的脚本名称存放在这一行的数据列中,例如表 8-15 中的 Order;使用标志 X 代表脚本结束。

在 SAFS 框架中,相应的处理如下:

```
'对文件中每一行
```

```
Select Case (sActType)
Case "G"
    Process…
Case "L"
    Log(sData)
Case "S"
    CallScript sData
Case "X"
    Exit
```

8.5.9 关键字驱动测试设计

为了实现关键字驱动测试,Robot 提供了测试设计工具,来帮助测试人员生成关键字驱动所需要的表。另外,测试设计工具通过使用数据库,能够在工具级别为测试重用提供支持。测试设计工具主要包括两方面的功能:供技术测试人员创建测试的静态结构;供业务测试人员创建测试的动态结构。

测试的静态结构要求和应用保持一致,以页面为单位。即应用中各个功能的层次结构是如何来安排的,就相应地在测试设计工具中,按照这种安排来建立静态结构,直到每个页面为止。这样来设计的好处是:

首先,静态结构和应用保持一致,将来应用发生变化,比较容易定位到静态结构中需要修改的地方;其次,建立静态结构,应用是什么样子,就建成什么样子,照搬即可,不需要很多的业务知识,比较适合于技术测试人员;最后,静态结构和应用保持一致,将来业务测试人员设计测试的动态结构时,能够方便地根据应用在静态结构中找到相应的页面。

图 8-9 是已经建好的静态结构的示例。



图 8-9 静态结构示例

在图 8-9 中，左边是和应用功能组织保持一致的树型结构。单击“集团理财”节点，可以在右边的上半部分看到此页面中的元素。页面上每一个元素都按照 Robot 技术框架的要求输入必要的信息，例如对象类型、对象标志、执行动作等。这些内容是由技术测试人员根据页面来输入的。如果不希望人工输入的话，那么也可以开发相应的工具去解析页面，来自动地生成每个页面的元素，或是使用 IBM Rational Functional Tester 的对象映射功能，由 IBM Rational Functional Tester 去页面上抓取对象来生成。

测试的动态结构和测试的要求有关。在创建测试用例的过程中，测试用例的每一个步骤，均是选自静态结构中的一个页面，将页面加入到测试用例中之后，还可以指定此次测试用例要测试页面上哪些元素。另外，在测试的动态结构中，还可以指定测试数据、验证点、数据关联等操作。当设计完成后就直接生成真正可以被 SAFS 框架所执行的表。

图 8-10 是已经建好的动态结构的示例。

测试准备

网银系统
企业网银
集团理财
登录
票据托管导航
票据托管
退出
公用数据池
日期
余额

测试项名称	类型	动作	属性字符串
提交	button	Press_Button	Index=1
出票人	editbox	Input_Editbox	Name=data
出票日期	editbox	Input_Editbox	Name=person

状态栏

图 8-10　动态结构示例

在图 8-10 中，左边是按照测试的要求组织起来的测试用例。单击“票据托管”这个测试用例，可以在右边的上半部分看到此测试用例的执行步骤，例如第一步是“登录”，第二步是“票据托管导航”，依次下来是“票据托管”和“退出”，这些步骤都是从静态结构中选出来的。当单击测试步骤中“票据托管”这个页面，在下方将此页面的元素显示出来，业务测试人员可以为每一个测试元素输入数据、指定数据关联、添加验证点等。

当业务测试人员设计好测试用例后，就可以将测试用例传递给 SAFS 框架，由测试技术框架解释并执行。

8.6 Rational Functional Tester 测试实践

IBM Rational Functional Tester(简称 RFT)是一款先进的、自动化的功能和回归测试工具,它适用于测试人员和 GUI 开发人员。在功能测试脚本的录制过程中,方便选择被测应用图形界面上的各种被测对象,进行参数化,通过生成新的数据池字段或从数据池中选择已存在数据字段,实现数据驱动的功能回归测试。

由于 RFT 和 Robot 同为功能测试工具,因此使用用法共同点很多。FRT 主要用于 GUI 测试,这里只重点介绍 GUI 测试所要用到的分层测试理念、对象识别技术和 ScriptAssurance 专利技术。

8.6.1 分层测试理念

基于 RFT 的自动化测试,都会采用 IBM 公司推荐的 ITCL 框架,这个框架采用了分层测试理念。

由于软件开发过程中,GUI 元素会发生一些变化,而操作流程的变动却不大。此时,由于业务流程脚本和原有的测试对象图(用于记录被测软件的被测对象)包含在同一资源当中,原有的 GUI 对象已经不能够使用,所以需要更新对象和业务脚本。当测试对象改动比较大、数量比较多的时候,测试脚本的维护开销会成倍增加。为了解决这样的问题,需要提出一种分层测试框架,把测试对象脚本和业务脚本分离开,在测试对象发生改动时,只需要重新维护测试对象图,对业务脚本不产生变更或产生少量的变更,从而大大节约脚本的维护开销。

图 8-11 就是典型的三层测试框架结构。

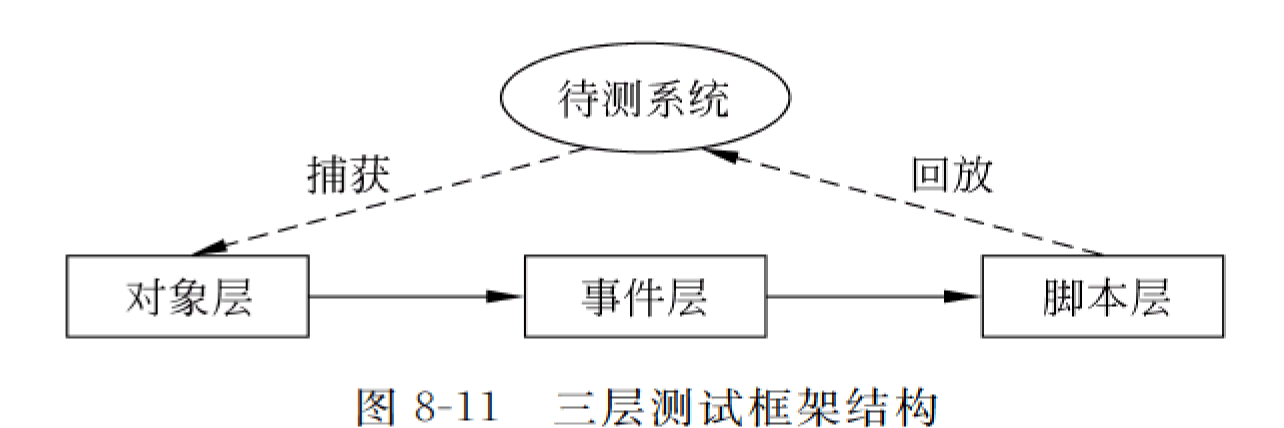
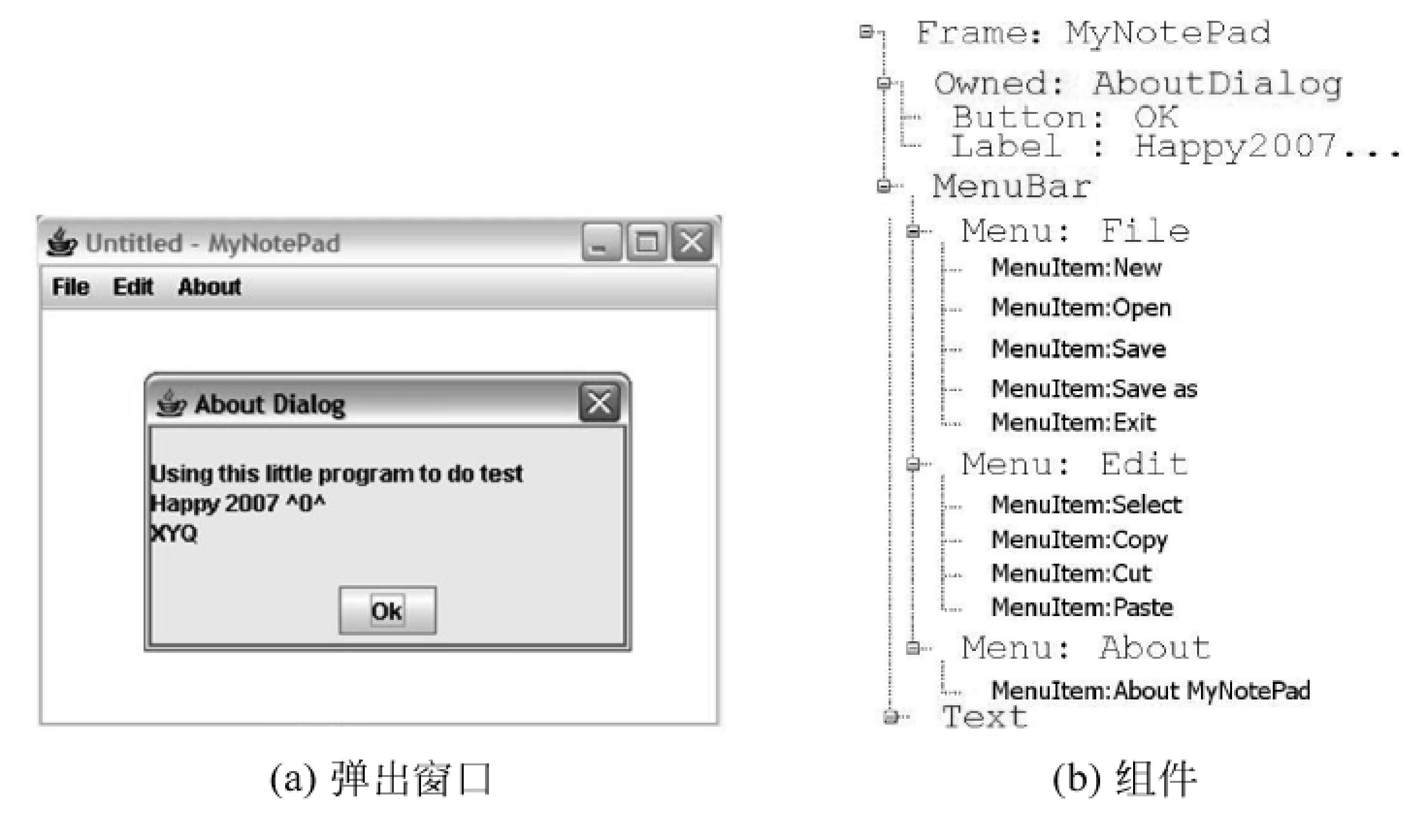


图 8-11　三层测试框架结构

在描述 GUI 对象表示时,本文将使用 RFT 来获得被测程序的所有对象。图 8-12(a)描述了 RFT 对捕捉到的所有被测对象的组织。是单击 MyNotePad 程序中 About 菜单的 AboutMyNotePad 菜单项时弹出的 About Dialog 对话框。图 8-12(b)描述了 RFT 中 MyNotePad 程序中的对象组织结构。

可以看到,MyNotePad 被组织成了一棵根节点,它是 MyNotePad 的树,它有 3 个子节点,分别为 About Dialog 对话框、MenuBar 和 TextArea。About Dialog 对话框有 2 个子节点,分别为 OK 按钮和一个 Label(存有 about 信息)。MenuBar 有 3 个子节点,分别为 File、Edit 和 About。TextArea 没有子节点。

可以看到，RFT 将 AboutDialog 对话框作为根节点 MyNotePad 的直接子节点。事实上，RFT 会将所有的弹出窗口存为根节点的直接子节点。使用这种组织结构，不可能从图 8-12 中了解到弹出窗口和其他 GUI 组件之间的关系。例如，AboutDialog 对话框应该在什么状况下弹出？为了解决这个问题，本文调整了被测程序的 GUI 对象组织结构，形成了一个新的结构图，如图 8-13 所示。这样的结构可以很容易地看出各个组件之间的依赖关系。



(a) 弹出窗口　　(b) 组件

图 8-12　树型结构

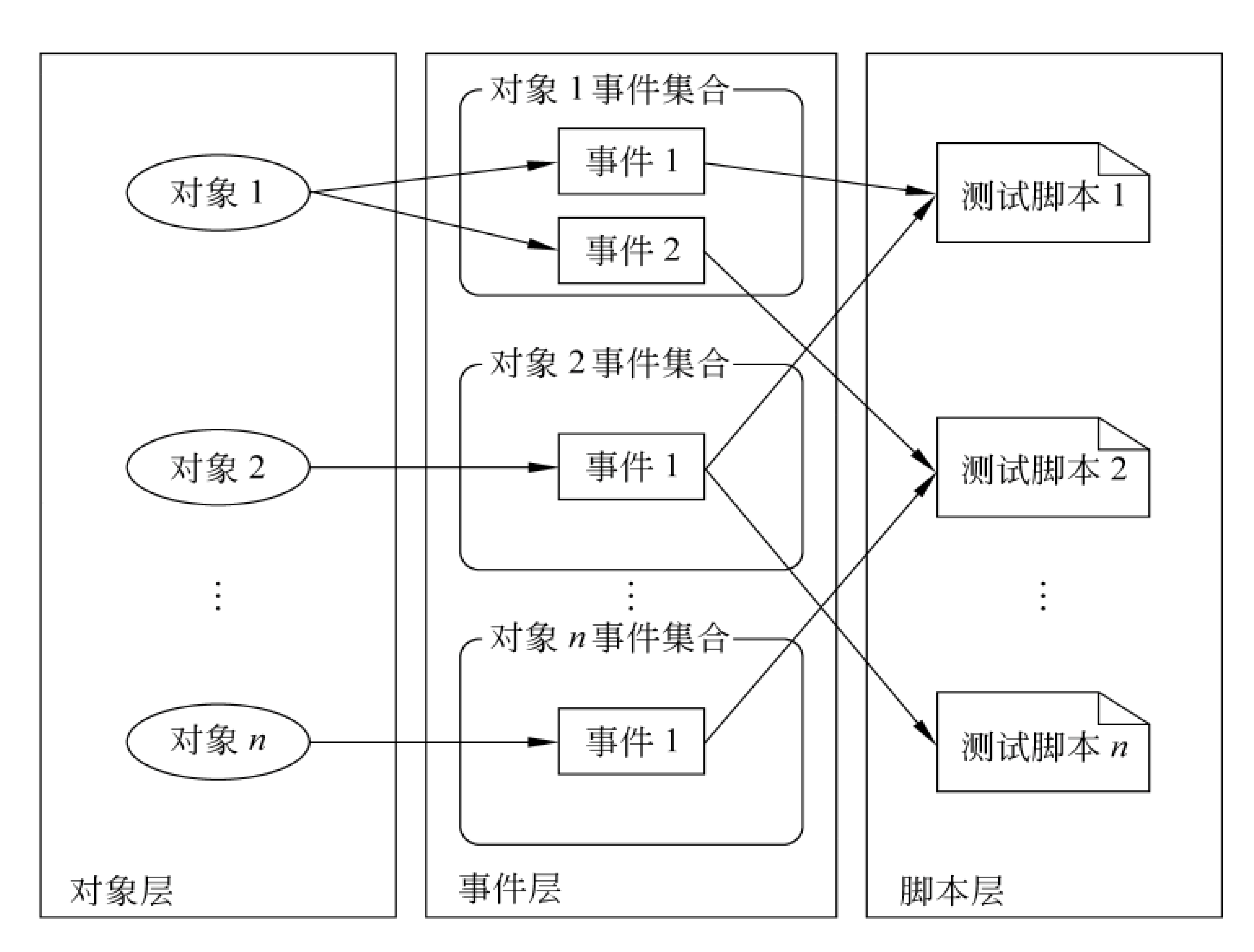


图 8-13　各个组件之间的依赖关系

对象层在这一层中，要将元素名与测试内部对象名独立出来。界面对象映射脚本，实现界面对象逻辑名与被测试软件真实界面对象之间的映射关联。界面对象映射脚本是自动化测试的关键，它可以使测试工程师和自动化测试脚本开发工程师进行分工，实现测试开发与软件开发的同步。在软件需求确定后，测试工程师就可以开发测试脚本，而自动化

工程师根据与测试工程师约定的界面对象的逻辑名来编写测试脚本。最后，测试工程师开发的测试脚本与自动化工程师开发的测试脚本，通过界面对象映射脚本，关联成一个有机的自动化测试脚本集。

事件层主要实施一些常用的测试操作，如文本内容录入、菜单选中、列表框内容选择、按钮单击、预期输出结果验证等。它是和对象相互对应的，每一个对象都应该有自己的事件集合。对常用的软件使用操作，可以针对每个类型控件设计一个通用脚本，控件识别ID 和相关联操作（单击、输入、选中等）作为该脚本的输入参数。对测试预期结果验证也可以使用类似的方法，设计针对特定控件的验证脚本。不同之处在于其输入参数是验证的属性和验证结果值。由于所有软件的使用和测试都是这些基本操作的组合，因而在不同的项目之间，重用脚本是一致的。不同的是测试所使用的测试数据从不同的数据文件内读取，因此，这些脚本可以在不同的项目中实施共享，实现一次编写，多处共享，减少脚本的数量，从而降低脚本的维护工作量。同时要将数据分离出来，数据可以来自文件或自动生成。因此要在事件层中为数据的输入和输出提供接口。

脚本层测试脚本实施对特定功能点和业务功能的测试，是一种针对特定的、被测试软件项目的脚本。其主要功能包括：从测试数据文件内读取测试数据，调用重用脚本和对象映射脚本将这些数据输入到被测试软件的特定对象中，并验证测试结果与预期输出的一致性，记录 Log 和 Bug 现象。它是由重用脚本依据一定的业务流程和特定操作流程组成。

8.6.2 对象识别

对象识别就是在 RFT 的对象模型框架下，得到被测程序的 GUI 对象。它是对象层开发中最核心的任务。常用的对象识别技术可以分为两大类：静态识别与动态识别。动态和静态方法各有优缺点，静态方法识别效率高、开发成本比较低，但是脚本的可维护性比较差，而动态方法刚好相反。

如图 8-14 所示，在 RFT 对象识别体系中，每个被测试的对象都被映射成为 TestObject 的子类实现。这些 TestObject 通过树型结构组织在一起，用以映射被测试对象中物件相互包含的关系。一个典型的例子，如图 8-15 所示，在这个例子中，整个 eclipse 的 Properties View 都映射成为一个 TestObject 的树型结构，这种结构就是专用测试对象图，如图 8-16 所示。



图 8-14 RFT 对象模型示例

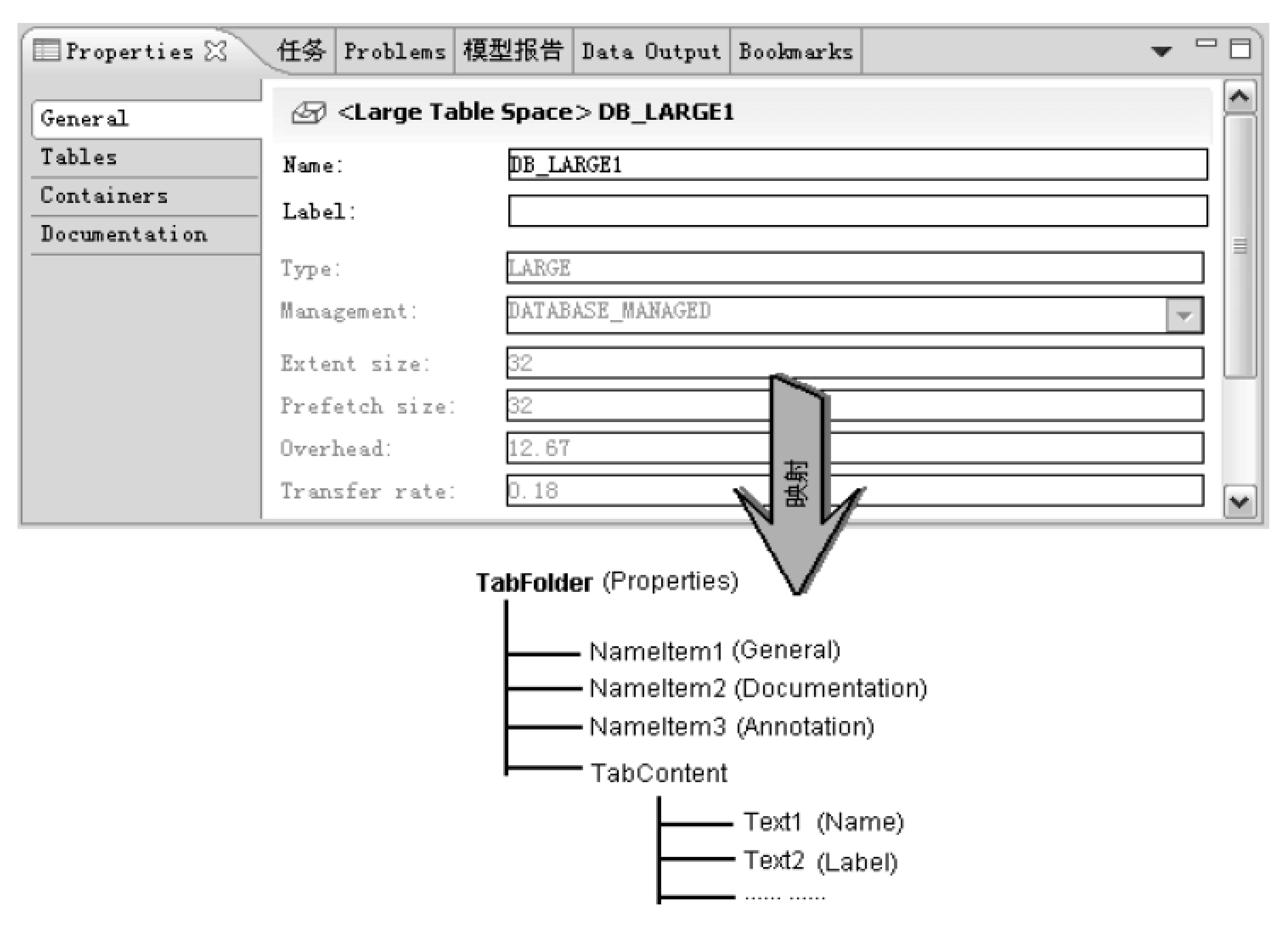


图 8-15　对象映射机制



图 8-16　专用测试对象图

其次这种可行性来自于 RFT 的 TestObject 类提供的丰富的对象属性。正是因为有这些属性的存在，测试人员才可能在动态遍历中寻找目标时，精确地判断相对位置关系和文字特征。RFT 中每个 TestObject 对象都可以使用 getProperties 和 getProperty 方法来获得主要的属性。在众多的属性值当中，bounds、class、text 是最常用的。bounds 参数可以返回当前被测试控件的坐标范围，class 参数返回被测试对象的 eclipse 类名称。text 参数返回被测试对象显示出来的文字。

RFT 会把被测应用程序（Application Under Test，AUT）中所有的元素都看成对象 TestObject，每个对象都由两部分组成：

- 一系列代表对象属性的键值对；

• 对象的层次结构。

脚本记录器将记录下来的对象，保存在该脚本的"专用测试对象图"中，而且是以树型结构保存。专用测试对象图不仅保存了对象本身的相关属性，连它和其他对象的相对关系也一并保存下来。

脚本回放时，RFT 利用"专用测试对象图"进行静态识别，可以从浏览器这个顶级容器开始，层层深入地定位到指定的某个对象。但这种呆板的对象查找方式也留下了很大的隐患：Web 页面里层次的结构千变万化，而这种变化对于对象的查找而言，有着致命的影响。在这种模式下要想适应页面层次结构的变化，只能重新录制对象，生成新的"专用测试对象图"。代价如此之大的维护方式，使得自动化测试几乎没有可行性。

因此必须将对象的识别同具体的"专用测试对象图"分开，实现对象的动态识别。页面对象都有一个共同的父类：com. rational. test. ft. object. interfaces. TestObject，而它的方法 find 正是用来在某个特定范围内查找满足条件的所有对象。借助它，可以对 SearchLotusLinkHelper 进行改造，使其与"专用测试对象图"不再紧密地耦合在一起。

改造后，回放过程中所需要用到的页面对象，都是在当前浏览器中即时查找得到的。通过目标对象的类型和某个属性值来定位目标对象，脱离了"专用测试对象图"中树型结构的约束。

RFT 采用以上对象识别技术，可以识别出大部分的 GUI 元素，但有时也会遇到 RFT 无法识别的 GUI 元素，事实上识别这类用常用对象识别技术无法识别的 GUI 元素占用了 RFT 脚本开发的大部分时间。

RFT 中一个非常重要的根接口是 IGraphical 接口，它定义了针对 GUI 元素的所有标准操作（单击、双击、拖曳等）。另外一个非常重要的根类是 GuiTestObject，它继承 TestObject 属性，并实现了 IGraphical 接口。常用对象识别技术中，GUI 元素都是被映射为 GuiTestObject 对象。它们在 RFT 对象模型中的位置如图 8-17 所示。灵活运用这些类、接口及其方法，能够极大地扩展 RFT 对象识别的功能。

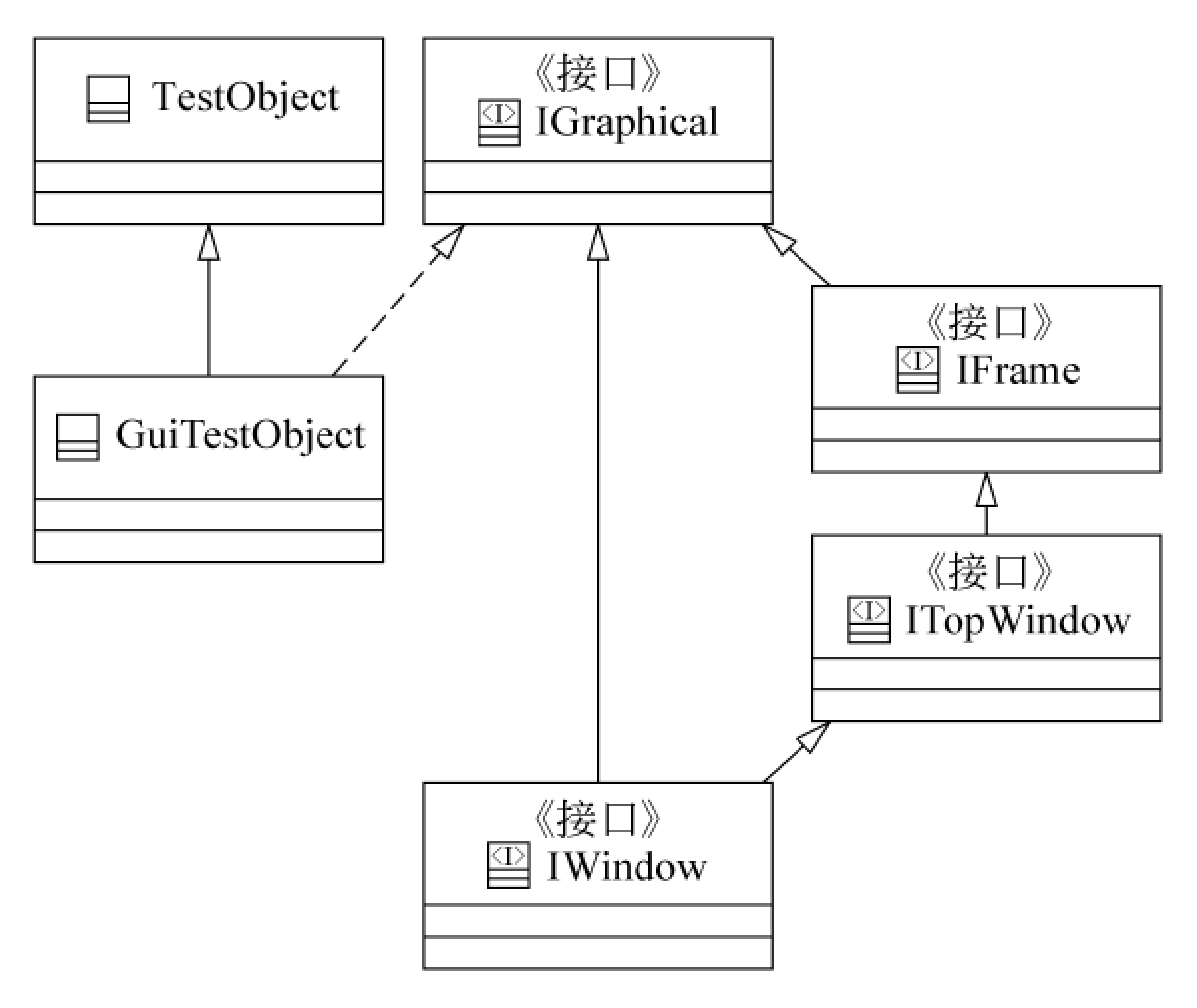


图 8-17　RFT 对象模型类图

8.6.3 测试对象和测试数据的维护

RFT 对测试对象的维护是通过对测试对象池的维护实现的。用户可以选择预定义的公用测试对象池，也可以选择在脚本录制过程中生成私有测试对象池，来维护通过录制自动获取到的或通过手工捕获得到的对象。测试对象池是 RFT 用来解决测试脚本在不同被测版本间成功回放的关键技术，它通过维护测试对象的各种重要属性，为测试脚本的执行和重用提供了重要保证。

RFT 由于采用了数据驱动技术，维护测试数据非常容易。当通过数据来驱动一个测试脚本时，脚本将使用变量作为应用的关键输入。通过使用变量，脚本能够使用来自外部的数据，代替应用测试中的文字值，图 8-18 显示了 RFT 使用数据池的数据驱动的测试。

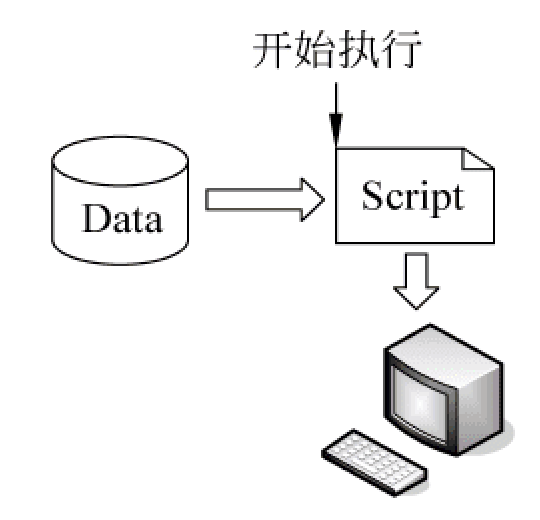


图 8-18 RFT 基于数据驱动的测试事例图

RFT 使用来自数据池的数据作为数据驱动测试的输入。一个数据池是相关数据记录的集合，在脚本回放时数据池能够为测试脚本提供测试数据。每一个测试脚本都有一个专用的测试数据池与之关联。RFT 提供通过创建一个新的数据池来创建一个共享的数据池，也可以将几个测试脚本关联到同一个共享数据池上。

数据驱动测试在数据域测试脚本之间放置了一个抽象的层次，这样可以消除测试脚本中的常量值。因为数据被从测试脚本中分离出来了，所以在 RFT 脚本维护的过程中可以做到：

- 在不影响测试脚本的情况下，修改测试数据；
- 通过修改数据而不是测试脚本来添加新的测试用例；
- 在多个测试脚本之间共享测试数据。

8.6.4 ScriptAssurance 技术

IBM 公司提供的 ScriptAssurance 专利技术，使测试员能够从总体上改变工具对测试对象变更的容忍度，在很大程度上提高了脚本的可重用性。ScriptAssurance 技术主要使用以下两个参数：脚本回放时，工具所容忍被测对象差异的最大门值和用于识别被测对象的属性权重。使用这种技术，测试员可以通过 Eclipse 的首选项设定脚本回放的容错级别，即门值，如图 8-19 和图 8-20 所示。

点击高级，能够看到各种具体的可接受的识别门值。

其次，测试员可以根据被测对象实际更改情况，在图 8-19 中修改用于回放时识别被测对象的属性及其权重。在测试脚本回访时，测试对象的识别分数将由以下公式计算得出：

```
int score=0;
for ( int i=0; i <property.length; ++i)
```

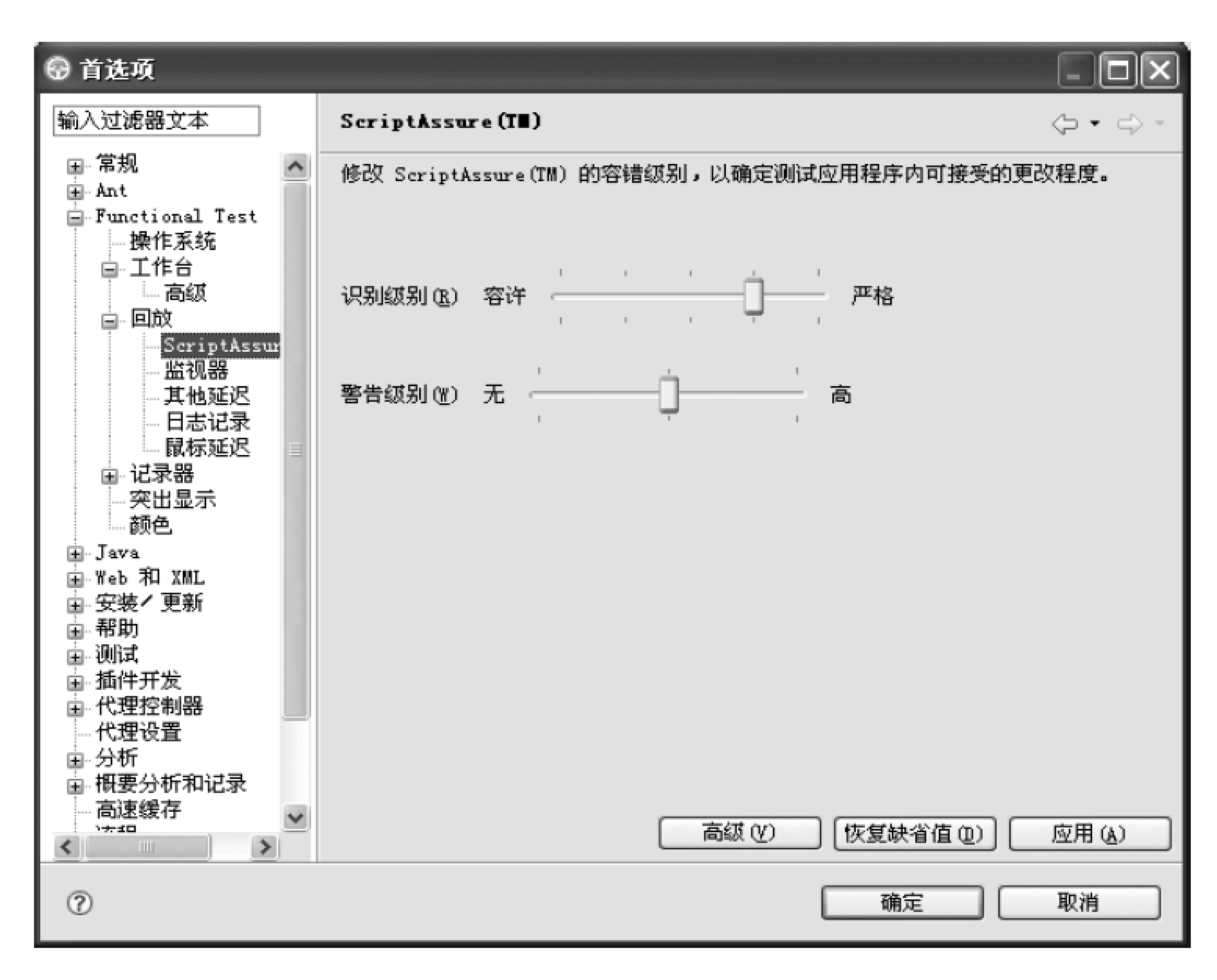


图 8-19　ScriptAssurance 容错级别设定



图 8-20　ScriptAssurance 门值设定

```
score+=(100-match(property[i])) * weight;
```

其中，match()将根据属性的符合程度返回 0～100 之间的值，完全符合返回 100，完全不符合返回 0。

测试脚本回放成功与否取决于“识别得分<识别门值”。通过这一技术，如图 8-21 所

示，通过设置恰当的 ScriptAssurance 门值和为用于识别对象的属性设置合适的权重，即使在两个回归测试的版本间测试对象有多个属性不同，对象仍有可能被正确识别，脚本仍有可能回放成功。这为测试脚本的重用提供了最大程度的灵活性。

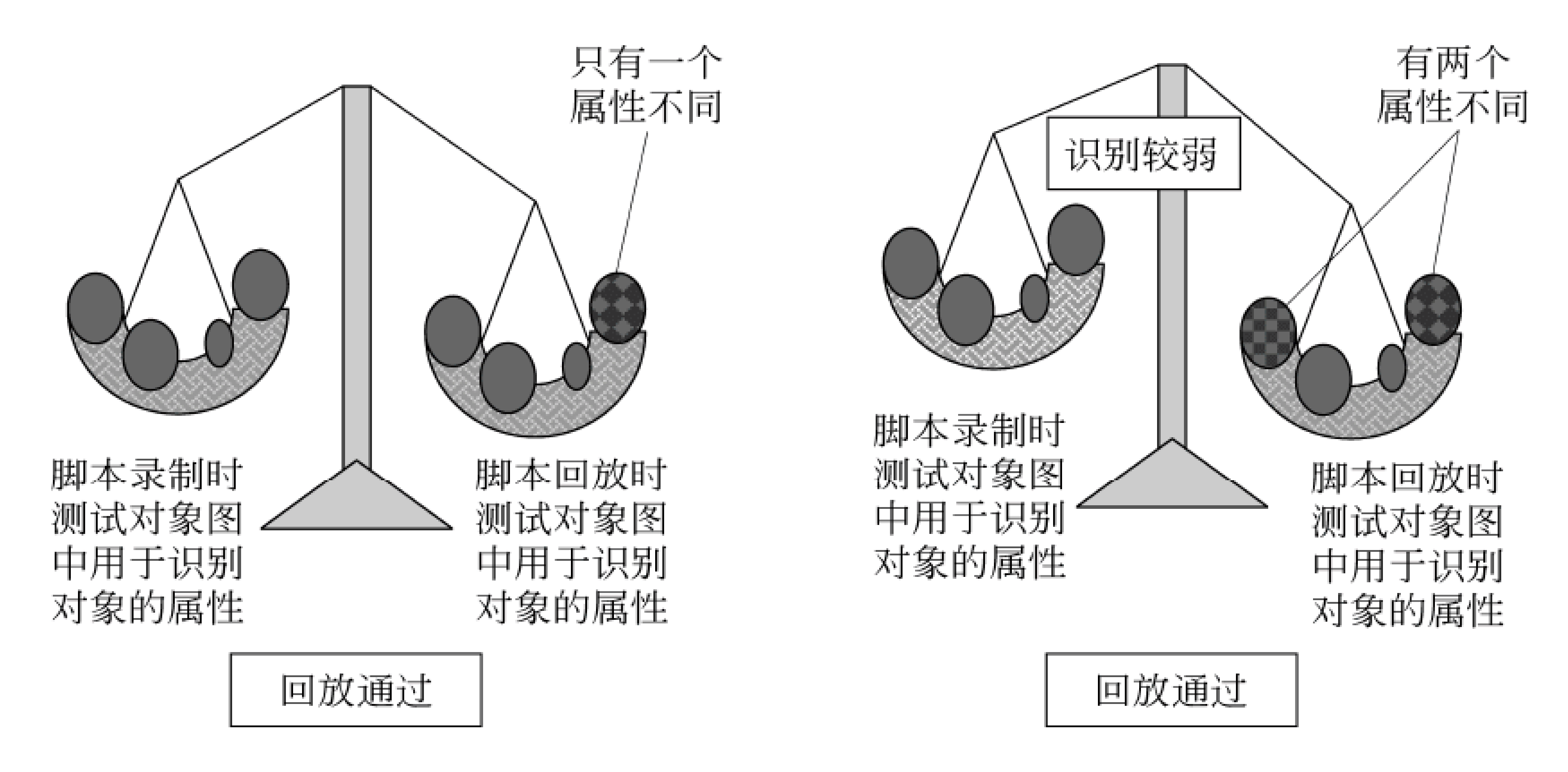


图 8-21　ScriptAssrance 技术保证脚本的重用

8.6.5　RFT 应用

RFT 的最大特色就是基于开发人员的同一开发平台（Eclipse），为 Java 和 Web 测试人员提供了自动化测试能力。

在 RFT 中实现测试脚本的过程和大部分的自动化测试工具一样，是基于录制的脚本生成技术。当完成测试用例设计后，只要在功能测试工具条上选择测试脚本录制按钮，就会启动测试用例的脚本实现过程。

在测试脚本录制过程中，测试员可以根据需要插入验证点和数据驱动的测试脚本，验证点是在指令中比较实际结果和预期结果的测试点，自动化功能测试工具正是通过它实现对被测系统功能需求的验证。

完成脚本录制过程以后，RFT 会自动生成用工业标准语言 Java 描述的测试脚本。基于 Java 的测试脚本，为软件测试员提供了更强大的编程和定制能力，测试员甚至可以通过在 Helper 类中加入各种客户化脚本，实现各种高级测试功能。RFT 具有基于向导（Wizards）的数据驱动的功能测试能力。在功能测试脚本的录制过程中，可以方便选择被测应用图形界面上的各种被测对象，进行参数化，通过生成新的数据池字段或从数据池中选择已存在数据字段，实现数据驱动的功能回归测试。

在生成测试脚本的同时，RFT 还能够帮助测试员在验证点中，使用正则表达式或使用数据驱动的方法，建立动态验证点。动态验证点用来处理普通验证点的期望值随着输入参数不同而发生变化的情况。通过简单操作，测试员就可以很方便地实现动态验证点的功能。此外，测试员还可以通过在验证点中使用正则表达式建立更加灵活的验证点，保证测试脚本的重用性。

使用 RFT 工具进行 Java 和 Web 应用系统测试时，使用基于 Java 的测试脚本语言保

证了测试脚本有良好的可重用性和脚本能力。此外，通过维护测试对象池，IBM 公司为测试员提供了不用任何编程，就可以实现测试脚本在不同的被测系统版本间的重用能力。

测试对象池分为两种：一种是公用测试对象池，它可以为项目中的所有测试脚本使用；另一种是私有测试对象池，它只被某一个管理的测试脚本所使用。在软件开发的不同版本间，开发员会根据系统需求的变化，修改被测系统和用于构建被测系统的各种对象，所以测试脚本在不同的版本间进行回归测试时经常会失败。因此，通过维护公用测试对象池，测试员可以根据被测应用系统中对象的改变，更新测试对象的属性值及对应权重，这样在不修改测试脚本的前提下，就能使原本失败的测试脚本回放成功。同时，为了方便测试员对测试对象池的修改和维护能力，RFT 还提供了强大的查询和查询定制能力，帮助测试脚本维护人员快速找到变化的测试对象，进行修改和维护工作。

8.7 小　　结

Robot 和 RFT 同属于 IBM 公司旗下产品，共同点很多，例如两者都具有广泛的环境支持和良好的灵活性，可以测试在几乎所有环境中被创建的应用程序，录制脚本也都很灵活方便，回放速度、数据池等很多方面都差不多，更重要的一点就是用法都很类似，使用方法都可以通用，这样可以减轻用户的学习负担。

当然区别还是有的，例如 Robot 使用 SQA Basic 脚本语言，而 RFT 使用的是 Java 或者 VB. NET。这与两者定位不一样，RFT 相对来说高级专业一点。SQA Basic 是足够简单易懂的语言，没有编程经验的测试人员也能较容易理解，而 Java 或 VB. NET 则常用在大型软件开发上，对开发人员要求高一点。

当然从以上解读可以看出 RFT 功能比 Robot 强大，例如对象识别能力，RFT 对象识别能力比 Robot 就有了很大的提高，主要是因为用到了对象库技术。而且 RFT 的文本编辑能力也比较好，因为 RFT 是基于 Eclipse 编辑器的，所以编辑功能比 Robot 大大加强。RFT 的日志记录能力也比较强，它可以形成 HTML、TEXT 日志或记录在 TestManager 里。而且除了文档，它还可以生成伪视频，捕捉在出错前的一个短视频。

总之 Robot 和 RFT 都是功能强大的测试工具，各有千秋，读者可以根据实际情况恰当选择。而且这两者还可以与 IBM Rational 整个测试生命周期软件的完美集成，真正实现了一个平台统一整个软件开发团队的能力。

习题与思考

1. 什么是系统功能测试？
2. 软件测试常用的功能测试方法有哪些？
3. 系统功能测试面临哪些挑战？
4. 什么是正则表达式？用什么表达式来匹配以空白符号间隔的数字（不包括这些空

白符号)?

5. 自动化功能测试面临哪些挑战?
6. 对表驱动进行简单介绍。
7. 什么是验证点?
8. RFT 提供了哪些验证点?对它们进行简单介绍。
9. 对 IBM Rational Functional Tester 进行简单描述。
10. 如何通过 Eclipse 的首选项设定脚本回放的门值?

第 9 章　性能测试

一则新闻：

官方网站 2007 年 10 月 30 日讯：今天上午 9 时，北京奥运会门票面向境内公众销售第二阶段准时启动。截至上午 11 时，各个销售渠道共售出门票约 9000 张，其中官方票务网站和中国银行各代售网点所售门票数量占 98%。

从今天上午的情况来看，公众购买门票的热情极其高涨。有些群众很早就来到中国银行排队等候；官方票务网站的浏览量在第一个小时达到 800 万人次，每秒钟从网上提交的门票申请超过 20 万张；票务呼叫中心热线从 9 时至 10 时的呼入量超过了 380 万人次。由于瞬间访问数量过大，技术系统应对不畅，造成很多申购者无法及时提交申请。为此，北京奥组委票务中心对广大公众未能及时、便捷地实现奥运门票预订表示歉意。

从这则新闻，读者可以计算出官方票务网站浏览量平均为 2200 次/秒以上，从网上提交的门票申请 200 000 张/秒以上。这样大的流量，是该网站开发商在需求设计时没估计到的，性能测试时也没有模拟出那么多用户的数据量，所以性能无法达到实际要求，因而造成了网站崩溃。

在软件系统日益复杂的今天，性能已经成为软件质量最重要的衡量标准之一，这点尤其体现在和 Web 相关的系统上。软件几乎无处不在，在给用户带来方便的同时，也对开发人员和测试人员提出了更高的要求。性能测试不但要求测试人员具备很强的技术能力，还要具备综合分析问题的能力。本节从性能测试的概念入手，重点是强化性能测试的基础知识。最后要求理解和掌握 IBM Rational Performance Tester 的使用。

9.1　性能测试基础

目前很少看到性能测试的准确定义，但是性能测试又似乎是涉及范围非常广泛的测试。压力测试、负载测试、强度测试、稳定性测试、健壮性测试、大数据量测试……都和性能测试有着密切的关系。

这里主要从狭义和广义两方面来讨论性能测试。

狭义的性能测试主要用于描述常规的性能测试，是指通过模拟生产运行的业务压力或用户使用场景来测试系统的性能是否满足生产性能的要求。

例如，以实际投产环境进行测试，来求出最大的吞吐量与最佳响应时间，以保证上线的平稳、安全等。性能测试是一种“正常”的测试，主要测试正常使用时系统是否满足要求，同时可能为了保留系统的扩展空间而进行的一些稍稍超出“正常”范围的测试。

广义的性能测试则包括压力测试、负载测试、强度测试、并发(用户)测试、大数据量测试、配置测试、可靠性测试等，它是一切和性能相关的测试的统称。下面分别介绍各类测试的主要内容和特点。

1. 压力测试

压力测试是对系统不断施加压力的测试，首先确定一个系统的瓶颈或不能接受用户请求的性能点，来获得系统能提供的最大服务级别的测试。例如测试一个 Web 站点在大量的负荷下，系统的事务响应时间，它何时会变得不可接受或事务不能正常执行。

压力测试的目的是发现在什么条件下，系统的性能变得不可接受，并通过对应用程序施加越来越大的负载，直到发现应用程序性能下降的拐点。压力测试和负载测试有些类似，但是通常把负载测试描述成一种特定类型的压力测试——例如增加用户数量或延长压力时间用以对应用程序进行压力测试。

2. 负载测试

负载测试是对系统不断地增加压力或增加一定压力下的持续时间，直到系统的一些性能指标达到极限，例如响应时间超过预定指标或某种资源已经达到饱和状态。这种测试可以找到系统的处理极限，为系统调优提供依据。

压力测试侧重压力大小，而负载测试往往强调压力持续的时间。在实际工作中，没有必要严格区分这两个概念。

压力测试案例

微软在开发 IE 4.0 的时候，有一个非常强的竞争对手，因此微软首席执行官要求必须保证 IE 4.0 做得非常好。为了测试 IE 4.0 的长期稳定性，微软开发团队专门设计了一套自动测试程序，一分钟可以下载上千个页面。他们使用这个测试程序，对 IE 4.0 进行了连续 72 小时的测试，也没有出现像内存泄漏、程序崩溃等任何问题。

压力测试可以帮助找到一些大型的问题，如死机、崩损、内存泄漏等，因为有些存在内存泄漏问题的程序，在运行一两次时可能不会出现问题，但是如果运行了成千上万次，内存泄漏得越来越多，就会导致系统崩溃。

压力测试时间要求：根据微软的实践经验，如果一个软件产品能通过 72 小时的压力测试，则该产品超过 72 小时后出现问题的可能性就微乎其微。所以，72 小时就成为微软产品压力测试时间的标志。

3. 强度测试

强度测试主要是为了检查程序对异常情况的抵抗能力。强度测试总是迫使系统在异常的资源配置下运行。例如：

- 当正常的用户点击率为“1000 次/秒”时，那么就要运行点击率为“2000 次/秒”的测试用例；
- 运行需要最大存储空间（或其他资源）的测试用例；
- 运行可能导致操作系统崩溃或磁盘数据剧烈抖动的测试用例。

强度测试是一种特别重要的测试，对测试系统的稳定性，以及系统未来的扩展空间均具有重要的意义。在这种异常条件下进行的测试，更容易发现系统是否稳定以及性能方面是否容易扩展。

疲劳强度测试是一类特殊的强度测试，主要测试系统长时间运行后的性能表现，例如7×24 小时的压力测试。

4. 并发（用户）测试

并发（用户）测试主要指当测试多个用户同时访问同一个应用程序、同一个模块或数据记录时，是否存在死锁或其他性能问题。几乎所有的性能测试都会涉及并发测试。在具体的性能测试工作中，并发用户往往都是借助工具来进行模拟的，这些用户称之为并发虚拟用户。

5. 大数据量测试

大数据量测试分为两种：一种是针对某些系统存储、传输、统计查询等业务进行大数据量的测试；另一种是与并发测试相结合的极限状态下的综合数据测试。如专项的大数据量测试主要针对前者，后者尽量放在并发测试中。此外，也可以把大数据量测试分为“运行时大数据量测试”与“历史大数据量测试”来进行测试用例设计。

6. 配置测试

配置测试主要指通过测试找到系统各项资源的最优分配原则。配置测试是系统调优的重要依据。例如，可以通过不停地调整 Oracle 的内存参数来进行测试，使之达到一个较好的性能。可以看出，配置测试本质上是前面提到的某些种类的性能测试组合在一起而进行的测试。

7. 可靠性测试

可靠性测试是在给系统加载一定业务压力的情况下，使系统运行一段时间，以此检测系统是否稳定。例如，可以施加让 CPU 资源保持 70%～90%使用率的压力，连续对系统加压 8 个小时，然后根据结果分析系统是否稳定。

这么多类型的性能测试看起来很吓人，实际上它们大多是密切相关的。例如，运行8 个小时来测试系统是否可靠，而这个测试极有可能包含了可靠性测试、强度测试、并发（用户）测试、负载测试等。

因此，性能测试是为描述测试对象与性能相关的特征，并对其进行评价而实施和执行的一类测试，如描述和评价测试对象的响应时间、吞吐量，以及操作的可靠性和限制等特征。一般可以使用被测系统的动态监测报告、响应时间及吞吐量报告、百分位图报告和各

种性能比较报告对被测对象进行性能评测。

当实施性能测试时绝不能割裂它们的内部联系去进行，而应分析它们之间的关系，以一种高效的方式来规划和设计性能测试。

9.1.1 应用领域

性能测试往往是为了实现下面的一个或几个目标：

- 判定软件是否满足预期的性能需求；
- 根据测试结果判定软件的性能表现；
- 查找系统可能存在的性能问题，如果有，则找出并加以解决；
- 发现一些应用程序在功能实现方面的缺陷；
- 对一些存在性能问题的系统，找出瓶颈并加以解决；
- 为用户部署系统提供性能参考。

通过分析性能测试的种种目标，不难总结出性能测试主要应用在以下几个领域中，下面分别予以介绍。

1. 系统的性能瓶颈定位

系统的性能瓶颈定位是性能测试最常见的应用领域。借助性能测试工具，可以在测试场景运行过程中监控系统资源、Web 服务器资源等运行数据，与响应时间进行同步分析，可以在一定程度上进行性能瓶颈的分析与定位。

2. 系统的参数配置

通过性能测试可以测试系统在不同参数配置下的性能表现，进而找出令系统表现更优的系统配置参数，为应用系统投产提供最佳配置建议。

实际上，常见的应用系统发生性能问题的重要原因就是操作系统、数据库、中间件服务器等的参数配置不当。

例如分配给 Oracle 的内存大小与系统自身的业务特点有极大关系，配置不同的数据库，性能表现就会不同；而即使在内存一定的情况下，SGA 的分配也会对性能产生很大的影响。因此，要通过测试以确定内存的最佳配置。

3. 发现一些软件算法方面的缺陷

一些多线程、同步并发算法在单用户模式下测试是很难发现问题的，只有通过模拟多用户的并发操作，才能验证其运行是否正常与稳定。

例如，在一次性能测试过程中，测试人员模拟多个用户并发时发现的一个明显的缺陷：测试对象是一个随机分配任务的模块，只要有用户申请，就会给用户分配任务。这个算法在单用户情况下调试没有任何问题，但是当多个用户同时申请任务时，就出现把同一任务分配给多个不同用户的现象。经证实，这个缺陷就是“同步算法”发生了问题而引起的。

4. 系统的验收测试

系统验收测试经常会验证一些预期的性能指标，或者验证系统中一些事务指标是否符合用户期望，这时就需要借助性能测试来完成验证工作。

随着用户对性能的重视，现在性能测试几乎是系统验收测试中必不可少的内容之一，甚至用户自己都要进行专门的性能测试来验证系统上线前的性能，以保证运行时的性能稳定。因此，性能测试在用户验收测试中越来越重要。

5. 系统容量规划

通过总结系统在不同硬件环境下的性能表现，可以为系统部署时提供非常好的参考。对于一些性能要求较高的系统，性能测试可以为硬件规划提供合理的参考数据，使用户在购买硬件时"有据可依"。例如同一系列机型：两个 CPU 支持 500 用户并发、四个 CPU 支持 800 用户并发，这些都是用户根据自身需求来规划硬件的重要依据。

6. 产品评估/选型

产品评估/选型是性能测试的又一应用领域。通过性能测试，可以对产品的软硬件性能进行更全面的评估，选出更适合自己的产品类型。

性能测试的应用领域越来越广，因此在实际工作中，性能测试往往会同时应用在一个或多个领域。对于软件性能测试设计人员，应该根据测试的具体应用领域、测试原则和目标来进行性能测试的规划与设计。

9.1.2 常见术语

本节将介绍一些性能测试中的常见术语，只有掌握这些基础的性能测试知识，才可以进一步开展测试工作。性能测试常见的术语主要有并发、并发用户数量、请求响应时间、事务响应时间、吞吐量、吞吐率、TPS、点击率、资源利用率等，下面分别予以介绍。

1. 并发

狭义的并发一般分两种情况。一种是严格意义上的并发，即所有的用户在同一时刻做同一件事情或操作，这种操作一般针对同一类型的业务。例如，在信用卡审批业务中，一定数目的用户在同一时刻对已经完成的审批业务进行提交（操作的不是同一记录）。还有一种是特例，即所有用户进行完全一样的操作，目的是测试数据库和程序对并发操作的处理。例如，在信用卡审批业务中，所有的用户可以一起申请业务，或者修改同一条记录。

另外一种并发是广义的并发。这种并发与狭义的并发的区别是尽管多个用户对系统发出了请求或进行了操作，但是这些请求或操作可以是相同的，也可以是不同的。对整个系统而言，仍然有很多用户同时对系统进行操作，因此，仍然属于并发的范畴。

可以看出，广义的并发是包含狭义的并发，而且广义的并发更接近用户的实际使用情况，因为对大多数系统而言，只有数量很少的用户进行"严格意义上的并发"。对于性能测

试而言，这两种并发一般都需要进行测试，通常做法是先进行严格意义上的并发测试，这种并发一般发生在使用比较频繁的模块中。尽管发生的概率不是特别高，但是一旦发生性能问题，后果很可能是致命的。严格意义上的并发测试往往和功能测试关联起来，因为只要并发功能遇到异常，一般都是程序的问题，这种测试也是健壮性和稳定性测试的一部分。

2. 并发用户数量

关于并发用户的数量，有两种常见的错误观点：一种是把并发用户数量理解为使用系统的全部用户的数量，理由是这些用户可能同时使用系统；还有一种比较接近正确的观点是把用户在线数量理解为并发用户数量。实际上，在线用户不一定会和其他用户发生并发，例如正在浏览网页信息的用户，对服务器是没有任何影响的。但是，用户在线数量是统计并发用户数量的主要依据之一。

并发主要针对服务器而言，是否并发的关键是看用户的操作是否对服务器产生了影响。因此，并发用户数量的正确理解是，在同一时刻与服务器进行交互的在线用户数量。这些用户的最大特征是和服务器发生了交互，这种交互既可以是单向传送数据的，也可以是双向传送数据的。

并发用户数量的统计方法目前还没有准确的公式，因为不同的系统会有不同的并发特点。例如 OA 系统统计并发用户数量的经验公式为：使用系统的用户数量×(5%～20%)。对于这个公式，没有必要拘泥于它计算出的结果，因为为了保证系统的扩展空间，测试时的并发用户数量都会稍大一些，除非要测试系统能承受的最大并发用户数量。例如：一个 OA 系统的期望用户为 1000 个，只要测试出系统可支持 200 个并发用户就可以了。

3. 请求响应时间

请求响应时间是指从客户端发出请求到得到响应的整个过程的时间。这个过程从客户端发送一个请求开始计时，到客户端接到从服务器端返回的响应结果计时结束。在某些工具中，请求响应时间通常会被称为 TTLB(time to last byte)，意思是从发送一个请求开始，到客户端收到最后一个字节的响应为止所耗费的时间。请求响应时间的单位一般为“秒”或“毫秒”。请求响应时间的分解如图 9-1 所示。

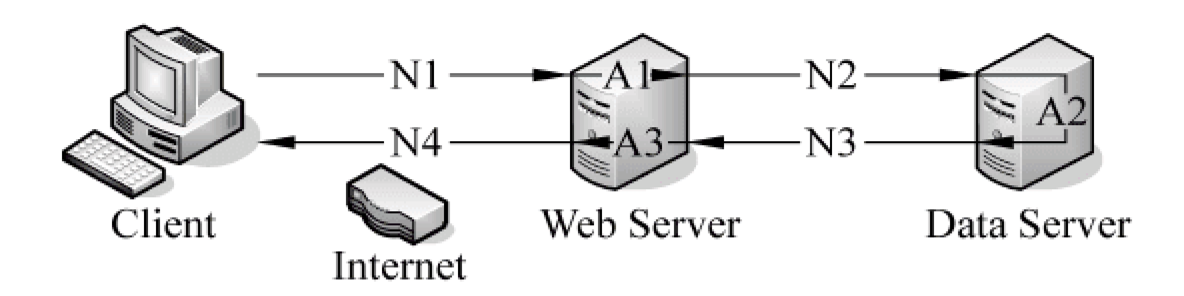


图 9-1 Web 请求过程分解

从图 9-1 可以看出，请求响应时间为“网络响应时间”和“应用程序与系统响应时间”之和，具体由 7 个部分组成，即(N1＋N2＋N3＋N4)＋(A1＋A2＋A3)。

4. 事务响应时间

事务由一系列请求组成,事务的响应时间主要针对用户而言,属于宏观上的概念,是为了向用户说明业务响应时间而提出的。例如,跨行取款事务的响应时间就是由一系列的请求组成的。事务响应时间和后面的业务吞吐率都是直接衡量系统性能的参数。

5. 吞吐量

吞吐量指在一次性能测试过程中网络上传输的数据量的总和。

6. 吞吐率(throughput)

吞吐率通常用来指单位时间内网络上传输的数据量,也可以指单位时间内处理的客户端请求数量。吞吐率一般是指吞吐量/传输时间,它是衡量网络性能的重要指标。

但是从用户或业务角度来看,吞吐率也可以用"请求数/秒"或"页面数/秒"、"业务数/小时或天"、"访问人数/天"、"页面访问量/天"来衡量。例如在银行卡审批系统中,可以用"千件/每小时"来衡量系统的业务处理能力。

7. TPS(transaction per second)

TPS 是每秒钟系统能够处理的交易或事务的数量。它是衡量系统处理能力的重要指标。TPS 是 LoadRunner 中重要的性能参数指标。

8. 点击率(hit per second)

点击率是每秒钟用户向 Web 服务器提交的 HTTP 请求数。这个指标是 Web 应用特有的一个指标。Web 应用是"请求-响应"模式,用户发出一次申请,服务器就要处理一次,所以"点击"是 Web 应用能够处理交易的最小单位。如果把每次点击定义为一次交易,点击率和 TPS 就是一个概念。不难看出,点击率越大,对服务器的压力也越大。点击率只是一个性能参考指标,重要的是分析点击时产生的影响。

需要注意的是,这里的点击不是指鼠标的一次"单击"操作,因为在一次"单击"操作中,客户端可能向服务器发出多个 HTTP 请求。

9. 资源利用率

资源利用率指的是对不同系统资源的使用程度,例如服务器的 CPU 利用率、磁盘利用率等。资源利用率是分析系统性能指标进而改善性能的主要依据,因此,它是 Web 性能测试工作的重点。

资源利用率主要针对 Web 服务器、操作系统、数据库服务器、网络等,是测试和分析瓶颈的主要参数。在性能测试中,要根据需要采用具体的资源利用率参数来进行分析。

9.1.3 性能测试的挑战

一般来说,在性能测试员进行系统性能测试的过程中,主要面临以下挑战:

• 性能测试脚本的能力：包括性能测试员构造各种复杂的性能测试场景的能力和测试脚本的扩展和维护能力。
• 测试脚本的参数化能力：性能测试总是要模拟大批量虚拟用户，对被测系统进行各种操作。因此测试脚本的参数化能力和上下文数据的关联能力，便成了性能测试员进行性能测试时要解决的基本问题。
• 构建各种负载模型的能力：准确模拟被测系统的真实负载情况，是确保性能测试有效、准确的前提。
• 被测对象的性能监控能力：它为性能测试员进行各种性能分析、定位问题和解决问题提供保证。
• 性能测试结果的分析能力：性能测试员需要使用各种报告和报表，对性能测试过程中的各种性能数据进行有效分析，做到正确认识被测系统的各项性能指标。

因此，优秀的性能测试工具，一定要满足以上各种性能测试能力要求，使得性能测试员在测试工具的帮助下，能够完成各种性能测试。

9.2 性能测试实践

市场上主流性能测试工具主要是 IBM Rational Performance Tester（RPT）和 HP Mercury LoadRunner，且 LoadRunner 的介绍比较多。这里以 RPT 为例予以说明。

RFT 是 IBM 公司基于 Eclipse 平台及开源的测试框架 TPTP 开发出来的最新性能测试解决方案，总体架构如图 9-2 所示。它可以有效地帮助测试人员和性能工程师验证系统的性能，识别和解决各种性能问题。它适用于性能测试人员和性能优化人员，用于开发团队在部署基于 HTTP 和 HTTPs 通信协议的 Web 应用程序前，验证其可扩展性和

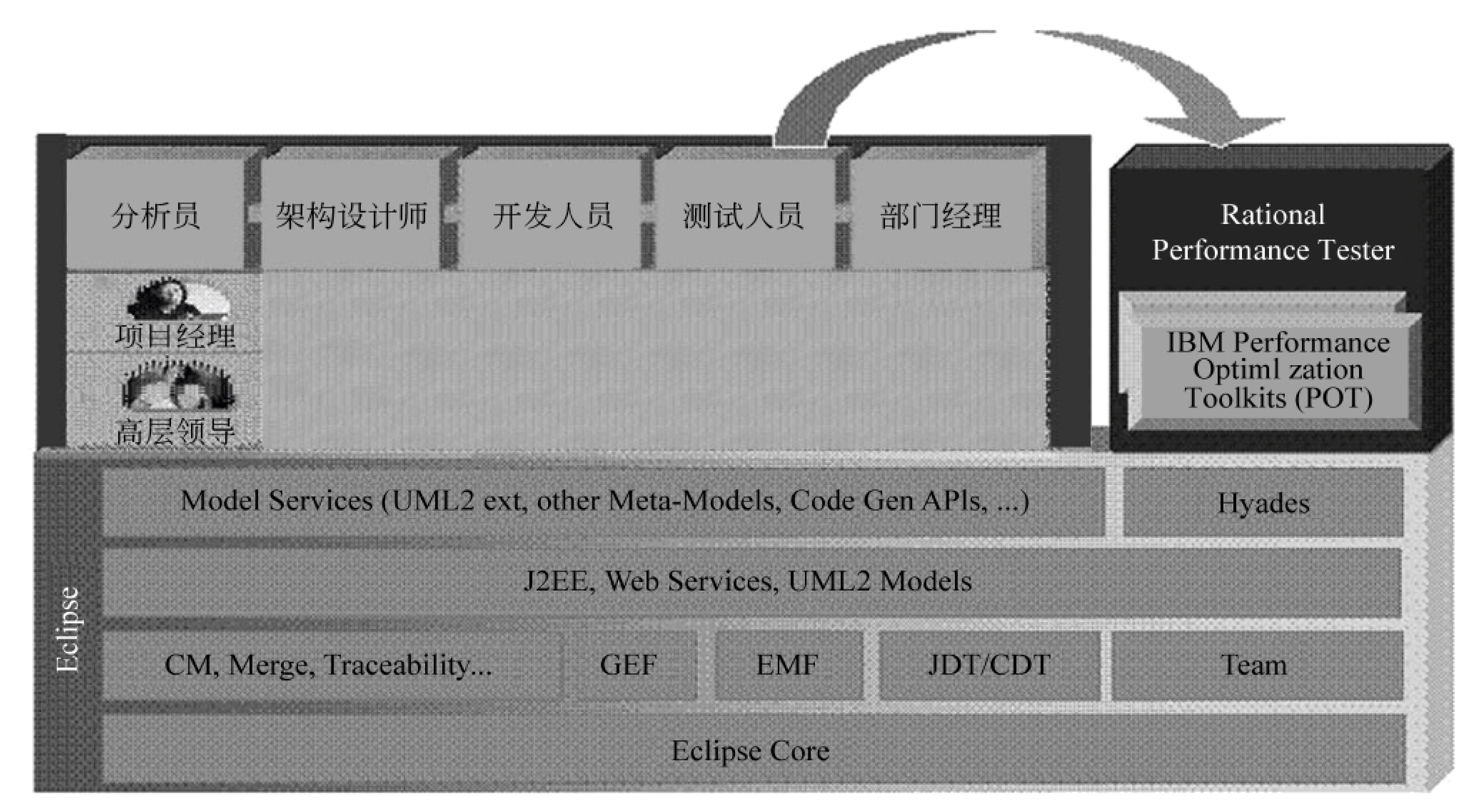


图 9-2　RPT 体系架构示意图

可靠性。在为性能测试员和性能优化人员提供了前面所提到的各种性能测试能力以外，它还提供了可视化编辑器，一方面可以使新的测试人员能在不需要培训和编程的情况下，即可快速上手完成性能测试；另一方面，也为需要高级分析和自定义选项的专家级测试人员，提供丰富的测试信息的访问和定制能力、自定义Java代码插入执行能力、自动检测和处理可变数据的能力。

测试框架 TPTP

8月3日，Eclipse基金会启动了新的软件测试工具项目TPTP，全称为Eclipse Test and Performance Tools Platform，并成立了专门的项目委员会，旨在为开放源码软件开发平台创建一个测试和性能框架。

Eclipse已经于2002年12月启动了一项名为Hyades的开放源码软件质量评估框架。该框架为数据模式、数据收集及执行、用户界面等制定了标准规范。此次启动的TPTP项目实际上是Hyades项目的进一步拓展。新增加的子项目主要包括：Hyades测试、Hyades追踪和Hyades监控。

据悉，该项目委员会由英特尔公司的软件开发工程经理Tyler Thessin领导，其他成员还包括SAP、Compuware和FOKUS等。

TPTP测试框架目前仍然在不断地发展，基于TPTP的框架，一方面能很方便地采用框架上已经提供的功能，例如DataPool、TPTP Log等，另外，因为TPTP本身基于Java架构，因此完全可以通过Java底层提供的强大的API来客户化代码满足各种不同环境下的不同需求。TPTP开放的架构、灵活的扩展方式和丰富的接口就是开源的力量。

此外，通过和IBM Rational的整个软件平台的完美集成，第一次为基于Eclipse的Web和J2EE应用系统的性能测试人员提供了和开发人员同样的操作平台，真正实现了一个平台、统一软件开发团队和性能测试团队的能力。

对于简单的性能测试，直接在一台机器上面安装RPT就可以进行所有的性能测试工作，但是对于复杂的、大用户量的性能测试，则需要在辅助机器上安装RPT Agent来产生虚拟用户，这样可以降低主控机器的资源占用率，以期获得准确的测试结果。RPT用RPT Agent模拟实际用户来和应用服务器进行交互，并对其产生工作压力。这些用户被称为虚拟用户。然而每个虚拟用户都会消耗一定RPT的系统资源，为了支持足够多的虚拟用户数目，一般需要部署多个RPT Agent来分担这些虚拟用户。同时有些虚拟用户执行的测试用例中需要某些软件或者Java库，这些库在远程主机上面，这时候，也需要在远程主机上安装RPT Agent来满足这些需求。

使用RPT对系统性能进行分析的过程包括四个步骤：

- 编写脚本。RPT提供了测试脚本的录制、定制和修正以及验证等功能。对于复杂场景的测试，则需要通过编写Java代码等方法来实现某些高级功能，例如不同的用户登录、随机数据、随机访问次数等。
- 测试调度。通过RPT，可以对多个测试脚本组合调度来进行性能测试。但是对于复杂的场景，则需要实现某些高级功能，例如按比例分配用户压力，实现用户平台期、数据池等。

- 测试运行。在完成编写脚本和测试调度后，可以开始运行整个测试。在运行过程中，可以通过不同的 Tab 页面查看性能数据。
- 测试结果分析。RPT 可以生成一些数据图表（如柱状图、折线图）来展现数据，对于用户来说，需要获得一些 Windows 资源数据（如 CPU 使用率、磁盘读写时间等）、响应时间、吞吐量、TPS（total transaction per second）等数据来进行分析。

下面就分析过程进行详细介绍。注意安装 RPT 会需要比较长时间，尤其是后面的更新程序步骤，一定要耐心地等待。等待更新完毕后，会提示安装完成。

9.2.1 脚本开发

RPT 脚本的开发过程通常都是采用"录制＋定制"的模式。首先通过对典型业务逻辑的录制，完成脚本中的基本业务的框架，然后针对录制结果，通过参数化、数据关联、增加逻辑控制等方式，加强脚本的适应性来满足特殊的业务需求。

1. 脚本录制/定制过程

脚本录制/定制过程有两个办法：一是直接生成面向过程的运行代码，二是录制结果经过"翻译"生成最终的运行代码。

直接生成面向过程的运行代码，开发者可以对最终运行的脚本进行直接的修改与调整。这种开发方式比较灵活，当然相应地对开发者的编程基础，尤其是 Java 语言的了解，有比较高的要求。

RPT 是录制结果经过"翻译"生成最终的运行代码，RPT 的脚本录制过程可以拆分成两步。如图 9-3 所示，第一步，RPT 使用位于代理控制器上的记录器（RPT Recorder on RAC），负责记录用户的所有 HTTP 请求（录制性能测试过程原始协议数据），生成一系列的跟踪记录文件（如.recmodel 和.rec 文件），文件记录用户与服务器的交互过程。第二步，当用户完成脚本的录制过程之后，RPT Test Generator 能够根据跟踪记录文件"翻译"一遍，生成最终运行的测试脚本。

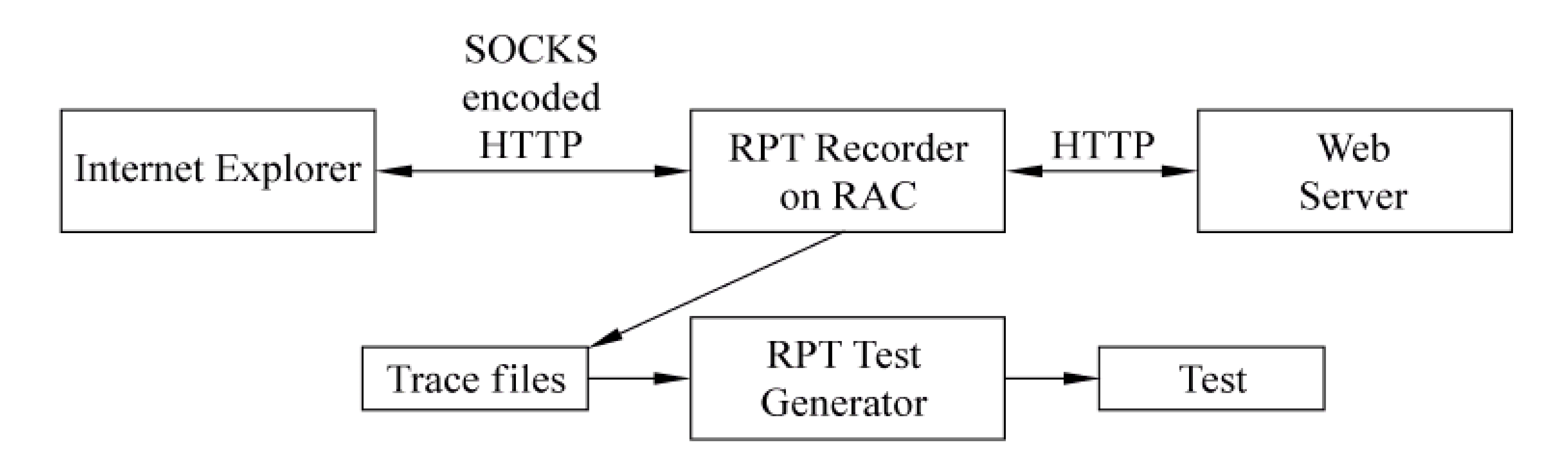


图 9-3　RPT 脚本的录制和生成架构

这种生成临时跟踪记录文件的好处是用户可以随时依据该跟踪记录文件生成新的测试脚本，然后再对脚本进行测试场景定制，而不用对同一个操作过程做多次录制操作。

在第二步里，RPT 测试生成器会对跟踪记录文件进行分析，生成测试脚本（.testsuite 文件）。性能测试员可以使用 RPT 提供的图形化测试脚本编辑器来观察测试脚本录制过

程中浏览器和 Web 应用服务器间所有的 HTTP/HTTPs 协议信息。在可视化的测试脚本编辑器中，性能测试员可以察看具体的消息协议数据，包括请求内容、响应头和响应内容，帮助测试人员理解测试脚本；也可以对指定的消息进行编辑、修改，添加定制的数据关联或进行脚本参数化；还可以加入定制的 Java 脚本，进行动态验证或控制测试执行逻辑。

2. 参数化

性能测试的主要任务就是模拟一定数量的虚拟用户，按照指定的负载模型对被测系统进行各种操作，完成测试。因此，性能测试脚本的参数化能力和消息上下文数据的智能关联能力，就会成为性能测试员工作中的一个重要任务。

录制业务流程时，RPT 生成一个包含录制期间用到的实际值的脚本。假设用户要使用不同于录制内容的值来执行该脚本时，就需要用参数替换已录制的值。这被称为脚本参数化。脚本的参数化可以简化脚本，同时增强脚本适用性。例如，如果需要搜索不同名称的图书，只需要编写一次提交函数，在回放的过程中，可以使用不同的参数值，而不只搜索一个特定名称的值。

如果用户在录制脚本过程中，填写提交了一些数据，例如要增加数据库记录。这些操作都被记录到了脚本中。当多个虚拟用户运行脚本时，都会提交相同的记录，这样不符合实际的运行情况，就有可能引起冲突。为了更加真实的模拟实际环境，需要各种各样的输入。参数化输入是一种不错的方法。

RPT 脚本参数化包含以下两项任务：

① 在脚本中用参数取代常量值。

② 设置参数的属性以及数据源。

参数化仅可以用于一个函数中的参量，不能用参数表示非函数参数的字符串。另外，不是所有的函数都可以参数化的。

RPT 的参数化过程同样简单，以替换用户登录密码为例来说明。首先，选中需要进行参数替换的请求页面，如图 9-4 所示，选中左侧的登录请求页面。在其右侧的 Test Data 中则显示与该请求页面相关的所有数据信息，脚本录制人员可以用其他值代替图 9-4 中的 password 变量。

3. 数据智能关联

数据关联类似于参数化，可以简化脚本，适应企业应用中需要动态数据的情况。默认情况下，RPT 在测试脚本录制和生成过程中，能够按照最佳实践经验，自动完成测试数据在不同消息间的智能关联，但是由于 HTTP 请求之间关联的复杂性，需要用户手动做一些数据关联。数据关联包含三个步骤，一是定义哪个录制的值需要被关联（替换）；二是定义数据源；三是定义被关联的数据与数据源之间的关联关系。

RPT 中如果需要自己定义关联，以更好的理解测试数据的来源，则在 HTTP 请求中的 URL 中或者 Data 中选择需要创建关联的部分，然后右键选择替换对象。其中替换对象可以是脚本中已经建立好的引用（这里的引用就是一种用户自定义的数据源），或者是 RPT 自带的数据源（如时间戳对象），或者是自定义代码。

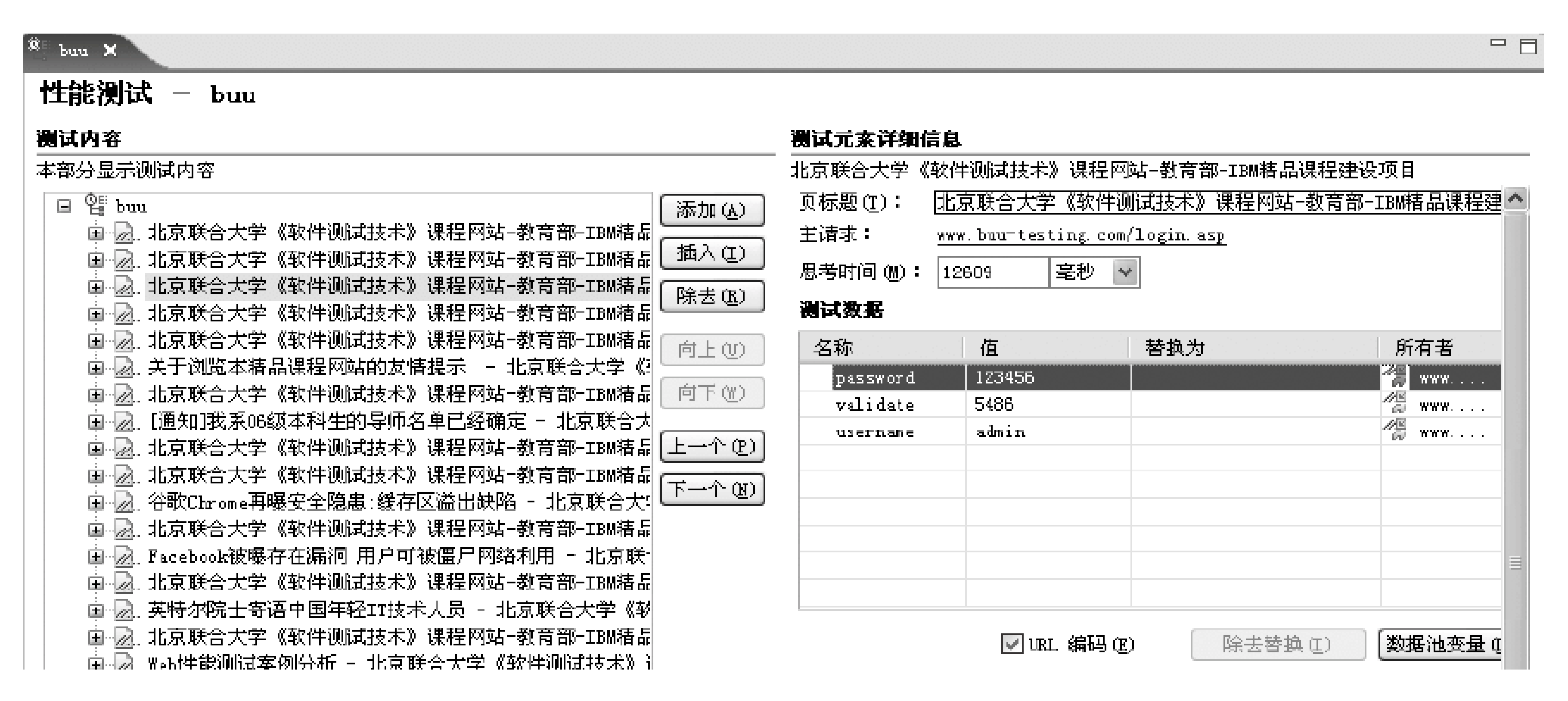


图 9-4　RPT 脚本参数化

RPT 会自动标识可能进行参数化的动态数据，测试员可以通过右键选取指定的数据，选择用数据池变量替换，从而实现测试脚本的参数化任务。RPT 使用绿底色标识指定的变量由数据池中读取。当然，在测试员可以使用数据池之前，首先必须在性能测试项目中创建所需的数据池，数据池中的数据可以从外部文件中导入，也可以在数据池的数据编辑窗口中进行编辑。

图 9-5 中右下部分是已经被关联的 URL，运行测试时该部分将由被引用的 URL 值来替换该 URL。

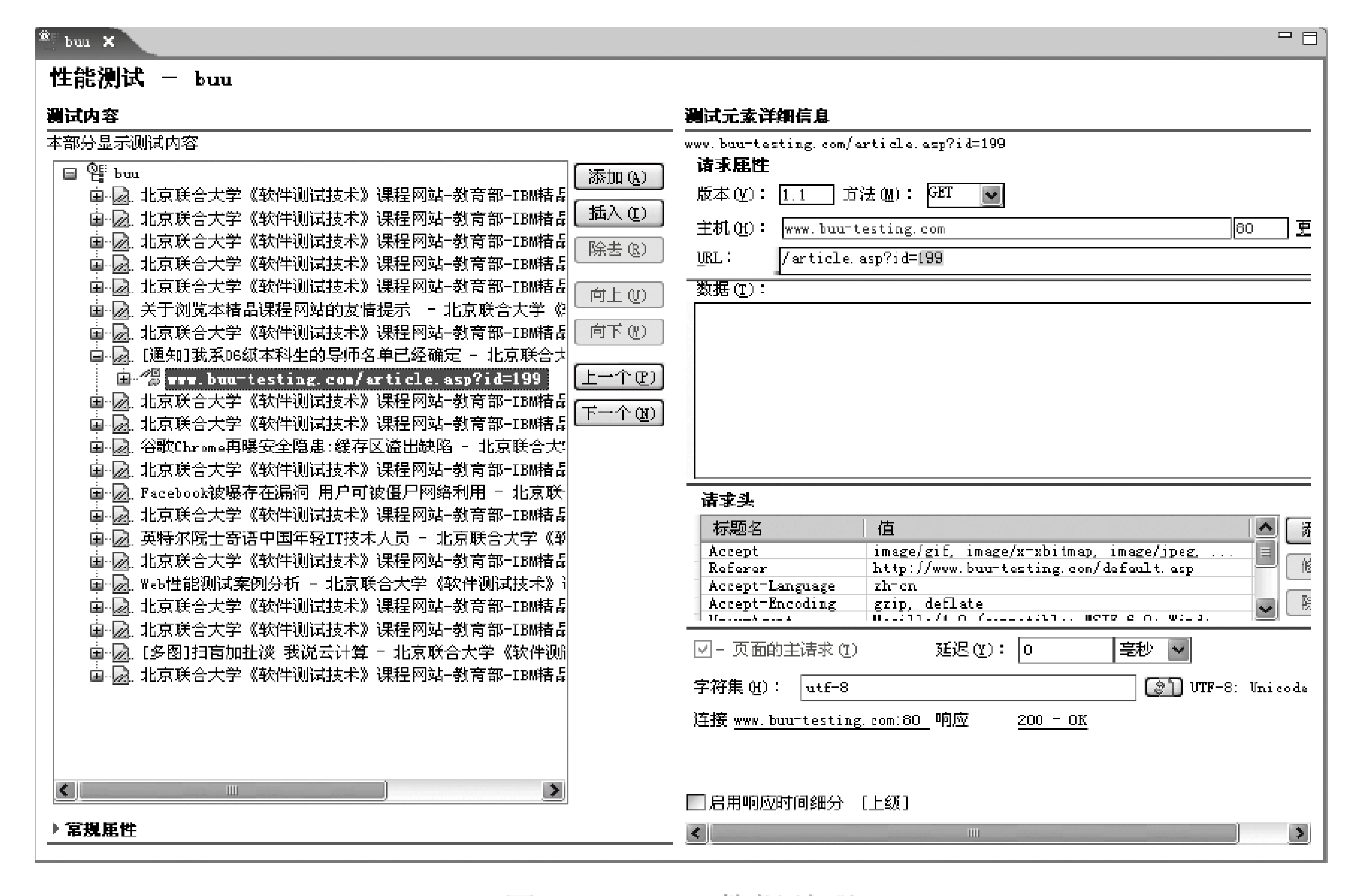


图 9-5　RPT 数据关联

4. 自定义代码

自定义代码(custom code)是 RPT 独有的概念。在测试脚本中添加自定义的 Java 代码，主要是为了实现对消息返回内容的验证、为其后的消息构造动态消息数据，或为了完

成如验证、加解密、日志记录等的特殊任务。RPT 通过内置 Java 代码执行引擎，提供在测试脚本中灵活插入客户化 Java 代码的能力。性能测试员可以通过右击的快捷菜单，方便地在测试脚本中添加定制 Java 脚本。

尽管在 RPT 脚本开发过程中，用户可以直接在 UI 层面达到对脚本的定制，但是这种定制能力毕竟有限。将定义好的自定义代码通过 UI 穿插到脚本之中，从而为 RPT 录制的脚本提供充足的扩展能力，来保证其灵活的定制性。自定义代码本质上就是一个 Java 类。自定义代码需要实现 com. ibm. rational. test. lt. kernel. custom. ICustomCode2 接口，并定义该接口中的方法如下：

```
public String exec(ITestExecutionServices tes, String[] args)
```

ITestExecutionServices tes 参数是 Test Container 中的一个实例，使用它可以访问 Test Container 运行一些服务。

String[] args 参数是定义自定义代码时定义传入的参数数组。

该方法的主体就是基于传入的信息进行业务逻辑处理代码，然后将处理结果（一个字符串）返回，其返回的字符串可以被后续的请求引用。自定义代码是一个纯 Java 的类，一般具有 Java 编程经验的人都可以根据业务需求编写自己的自定义代码。

5. 数据池

上面看了如何录制脚本，但是有些复杂的流程不可以仅仅靠录制脚本就能实现的。例如登录，假设需要有 45 个用户登录，那么不同用户登录提交的用户名和密码就应该是不同的，RPT 提供了数据池引用以及变量替代的功能，即将一个数据文件作为参数值赋给一个参数，就可以满足这样的需求，如图 9-6 所示。



图 9-6 RPT 数据池

在 RPT 中，用户只能按顺序从数据池中读入测试数据。RPT 的数据池是以 XML 格式存储的，并且在测试开始时，将数据池中的所有数据都加载到内存中，这样的实现模式不利于测试中使用大数据量。对于用户来说，有时候需要生成一些随机数据，而数据池的访问是顺序的，那么就需要用代码来实现这样的功能。灵活的 Custom Code 功能可以弥补这方面的不足。对于大量的测试数据，可以通过自定义代码来实现 On Demand 的数据读取和加载。

6. 流程控制

RPT 的流程控制操作可以通过用户交互界面轻松进行，它提供了灵活的流程控制模式，包括 IF 条件控制结构和 LOOP 循环结构。

（1）IF 条件控制结构

在 RPT 脚本中，可以将一部分连续的页面或者 HTTP 请求放到一个 IF 条件中去，然后由判断条件来确定 IF 结构中的页面或者 HTTP 请求是否被执行。其判断条件可以是 RPT 自动参数化后的参数，也可以是 Custom Code 的返回值或者是数字、字符串等。图 9-7 是添加了 IF 条件后的脚本，包含了为 IF 语句设置判断条件的配置界面。

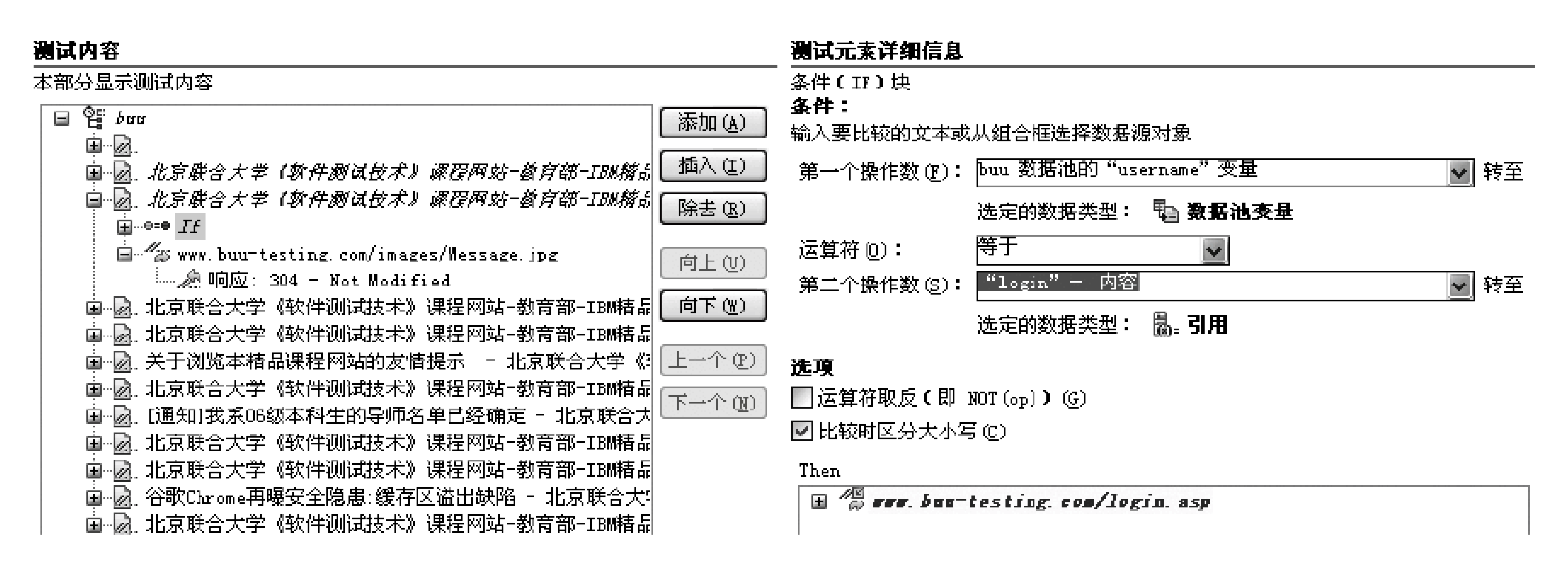


图 9-7　RPT IF 条件控制结构

同样，RPT 也支持 IF-ELSE 的结构。

（2）LOOP 循环结构

RPT 中的另外一种流程控制结构就是 LOOP 结构，如图 9-8 脚本中，将所有的页面放到了一个 LOOP 结构中，然后可以通过指定循环次数来确定其中的脚本循环执行多少次。

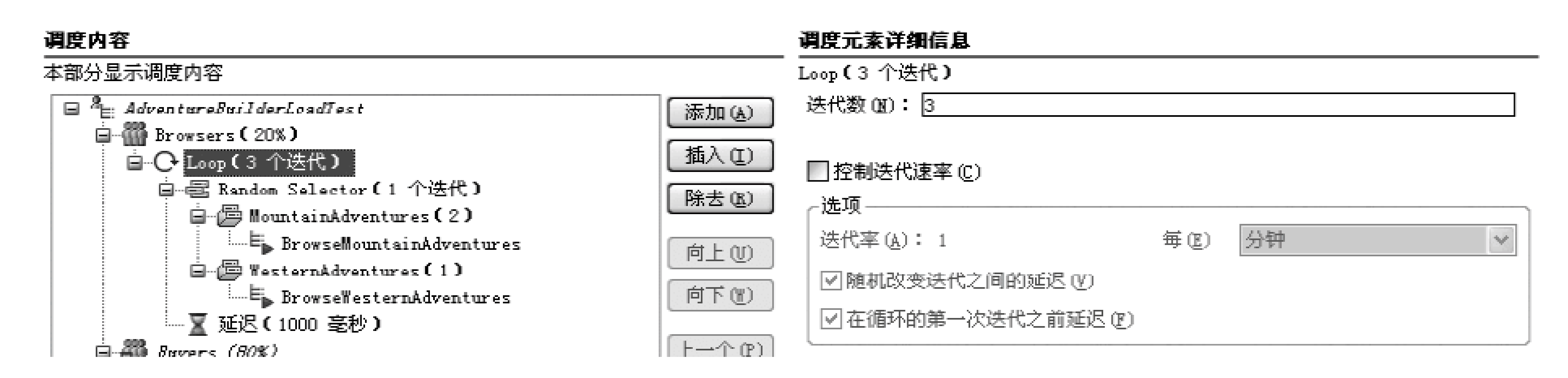


图 9-8　LOOP 控制结构

7. 全局信息

这里所说的全局信息，实际上是脚本运行时，RPT 产生的内部数据，例如：当时的运行时间，Vu 所在用户组组名，迭代编号等。RPT 中的全局信息，存放在 IDataArea 对象中。IDataArea 对象包含三个方面的信息，分别是“Test Data”、“Virtual User Data”、“Engine Data”，这些信息都可以通过 Custom Code 来获得。

Custom Code 的实现需要继承 ICustomCode2 类，并实现该接口的核心方法“public String exec(ITestExecutionServices tes,String[] args)”，该方法的第一个参数就可以获得 IDataArea 对象，然后获得全局信息。同时用户也可以向 IDataArea 对象中添加信息，提供给测试脚本的其他地方使用。

8. 错误控制

RPT 中，当脚本运行出现错误，脚本将继续执行，可能后续会出现很多错误。

RPT 在运行完一个测试之后，会产生相应的测试日志，如果在测试过程中发生任何错误，RPT 会以“Message”的形式提示出该请求发生错误。如图 9-9 中的测试日志中被选中的 Message 表示该 HTTP 请求在引用前面的关联值时发生错误。

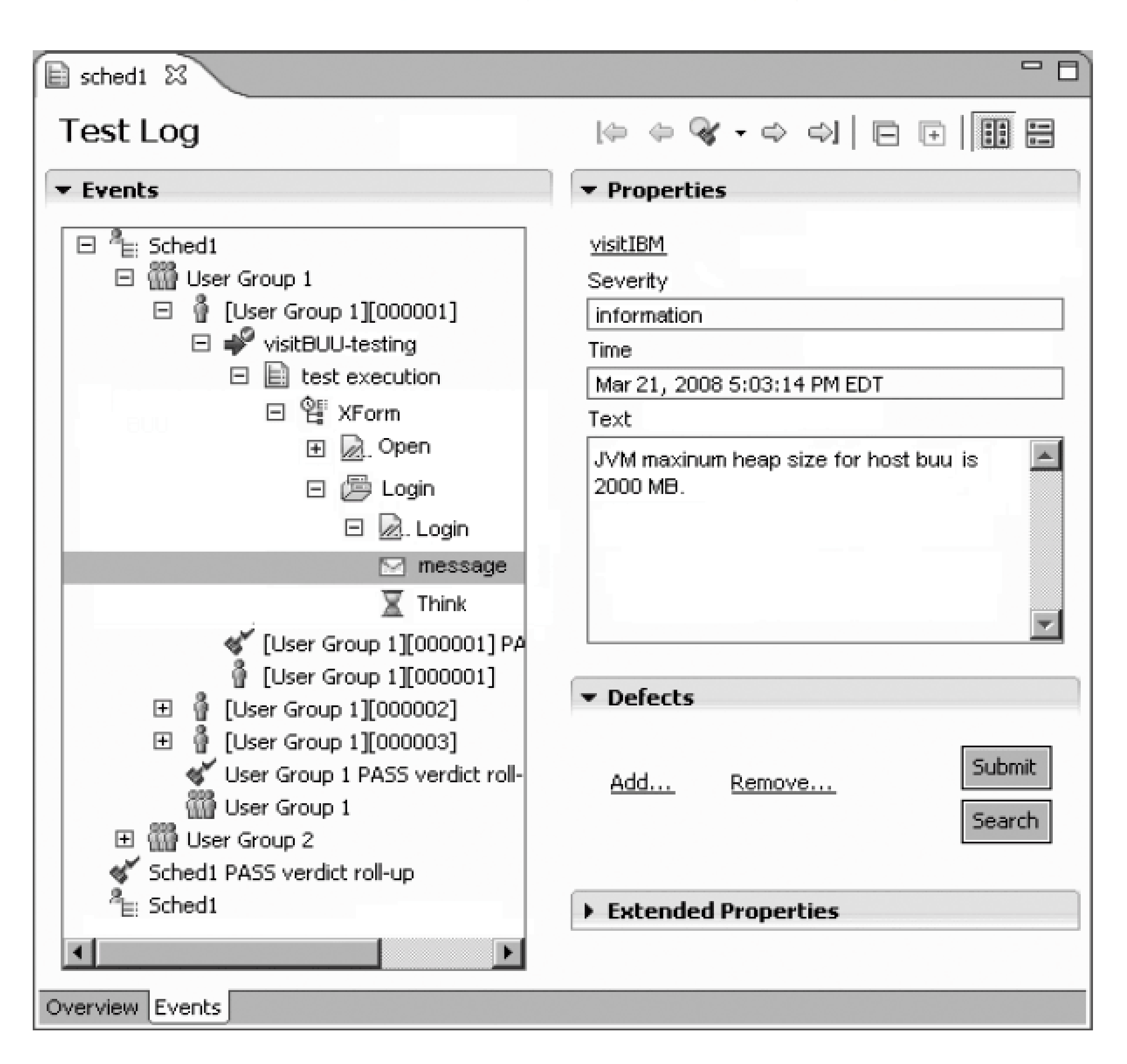


图 9-9　RPT 错误日志

验证脚本记录是否成功可进行测试脚本回放，如果回放测试脚本后，在“性能报告”界面的“总体”TAB 页中显示“完成”，并且界面中的柱状图都到达 100，“错误日志”视图内没有错误提示，则说明测试脚本回放成功了。

9. 同步点

同步点是 RPT v7.0.1 中新增的一项高级功能，为性能测试增加了灵活性。在性能

测试执行过程中，由于各个虚拟用户间的思考时间、页面响应时间等的个体差异，测试脚本中的各段操作时间也会不同。有时我们需要在所有虚拟用户同时达到某一点后执行某一操作。在这种情况下，借助同步点就可以轻松地达到这个目的。同步点可以让先达到这一点的虚拟用户暂停下一步动作，处于等待状态；当所有虚拟用户都到达同步点时才按照设置并发或按某一时间交错执行下一动作。在 RPT v7.0.1 中，同步点可以应用在性能测试脚本或测试调度中。但由于同步点主要用于控制脚本的执行，所以在测试调度中的应用要更多一些。

10. 优化测试脚本

当录制完一个基本的用户脚本后，在正式使用前还需要完善测试脚本，增强脚本的灵活性。一般情况下，通过以下几种方法来优化测试脚本：插入检查点、设置动态数据、参数化输入。下面详细介绍参数化如何设置，其他只作简单介绍。因为功能确认不是性能测试的首要目标时，就需要谨慎地向测试中引入其他形式内容的确认，以确保精确的回放。

（1）插入检查点

右击页面/页面请求/页面响应，在快捷菜单中选择启用相应的检查点。RPT 提供了三种检查点：

① 页面标题 VP：对预期标题大小写敏感。

② 响应代码 VP：设置响应代码 VP 后，在每个页面请求的响应下面增加一个“响应代码验证点”的文件夹。如果匹配方法选择“模糊”，那么记录测试时的响应代码为 200，在测试执行时，响应代码为 201、202 等均不会报错。

③ 响应大小 VP：设置响应代码 VP 后，在每个页面请求的响应下面增加一个“响应大小验证点”的文件夹。

（2）设置动态数据

在 RPT 里，可以通过数据池获得动态更新的数据。数据池把在记录过程中所捕获的每个单独的数据以一组测试运行中的数据值做替换。数据池的目的是通过为每一次测试提供唯一的数值，以确保回放的真实性。

创建数据池的步骤如下：

① 在“测试导航器”中选择需要创建数据池的项目，在快捷菜单中选择“新建”→“数据池”，弹出“新建数据池”对话框。

② 选择数据池所在的项目，输入文件名，如果想创建空的数据池，则直接单击“完成”键；如果需要将数据文件中的数据导入，那么就单击“下一步”键，选择需要导入数据的 csv 文件。如果欲导入的 csv 文件第一列是正常的数据，且第一行没有列名时，那么在导入 csv 文件的界面中，“第一行包含变量名和建议类型”和“第一列包含等价类”选项，均取消选中项。

打开数据池文件，“概述”Tab 页显示数据池的一般信息，“数据表”Tab 页显示导入的数据。这里第一列是等价类信息，在性能测试中，这列的内容不需要考虑。可以通过右击的快捷菜单进行数据的维护操作。

替换数据的步骤如下：

① 选择需要使用数据池替换的页面(页面里的页面请求变绿色)。

② 在“测试数据”部分单击“数据池变量”，在弹出的“选择数据池列”对话框中，单击“添加数据池”，弹出“导入数据池”对话框。

③ 在“导入数据池”对话框中，“匹配的资源”部分会显示所有目前打开工作空间中所有未被关联的数据池文件，选择需要关联的数据池文件，单击“选择”，返回到“选择数据池列”对话框。

说明：

① 在“导入数据池”对话框中，数据池的“打开方式”有如下三种：

- 共享(每台机器)：每台机器的虚拟用户从数据池的公共视图上取数据，并按照first-come-first-served 机制顺序把数据分配给虚拟用户。虚拟用户用迭代方式将从不同的行取数据，他们取的数据不可预知。
- 私有：每个虚拟用户从数据池的私有视图上取数据，并且使用相同的顺序将数据行分配给虚拟用户。
- 分段(每台机器)：每台机器的虚拟用户从数据池的分段视图上取数据，并按照first-come-first-served 机制顺序把数据分配给虚拟用户。分段是性能调度如何根据机器分配虚拟用户来计算的。例如，如果一个调度分配 25% 的用户给用户组 1，75% 的用户给用户组 2，并分别分配这些用户组在机器 1 和机器 2 上执行。为了防止虚拟用户取重复的数据时，这是一个比较好的办法。

② 如果想要在数据池数据用完后测试提示失败，那么可以将“到达最后一行时回绕”选项取消选中。

③ 在“选择数据池列”对话框中，选择需要使用的列，单击“使用列”，返回到工作台，此时会发现被数据池列替换的变量变成绿色。

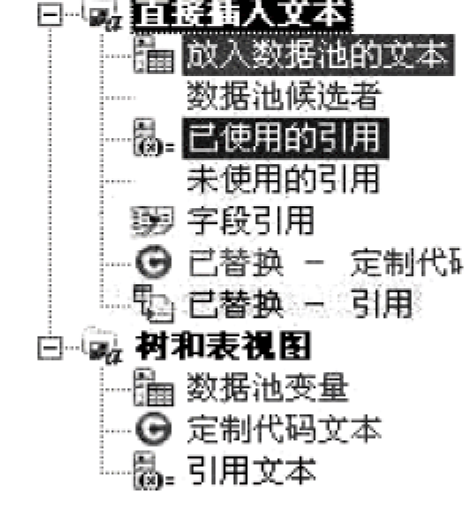


图 9-10　选择数据池列

使用 RPT 记录器，已经捕获了性能测试。在编辑器中测试，以树型视图显示。通过两种方法，可以优化这些测试：第一种，添加验证点以保证准确的回放。在回放过程中，如果任何包没有返回与在记录中被捕获的代码相同的响应代码，那就是测试失败，将会在事件日志中得到一个通知。第二种，数据池中记录输入的不同的登录用户信息。这将保证一个更真实的、不同虚拟测试器为它们的登录使用不同账号的回放。

9.2.2　场景构建与配置

脚本只是定义了某些用户的操作步骤，而一个场景则包含了有关如何模拟实际用户的所有信息，而构建场景，则必须考虑建立什么样的性能测试负载模型。

性能测试的关键是能够通过测试工具准确模拟被测系统在生产环境运行时的真实负载情况。在进行性能测试前，一般会由性能测试员和用户代表一起，根据性能测试计划中

指定的测试目标，制定测试用例，完成对应的负载模型分析，以便正确执行和实现性能测试目标。一般情况下，性能测试员使用“负载分析文档”来确定性能测试负载模型中要使用的各种变量，并定义变量值。通过它们来确定被测系统在生产环境中运行时，涉及的各种负载角色特征、每种角色要执行的最终用户业务功能（用例及其执行流程与条件）和对应工作量和容量，以便最恰当地模拟最终用户的负载情况。此外，负载模型中还应确定负载模拟持续的时间间隔、测试期间要改变的任何因素或变量，以及测试结果的评测方法。

进行负载模型分析的关键，在于找出被测系统的主要角色，以及主要角色所进行的关键任务，从而从总体上了解被测系统是如何被各种不同用户使用，以及在怎样的负载情况下工作的。在进行性能测试前，性能测试员可以通过以下手段获得系统负载模型中的各种变量。

① 从最终使用人员那里获得操作情况，如经常进行的业务类型、业务操作的频率。

② 根据系统日志，可以获得每日所进行的各种业务类型和业务量。

③ 通过和系统测试人员、操作人员、系统架构师充分沟通和配合，对被测系统作如下分析：

- 定义系统主要角色，它在 UML 中被称作主角（Actor），确定每个主角的属性和工作简档，确定哪些能够唯一标识被测系统最终用户的各种特征的属性和变量（如打字速度、思考时间以及反复出现的因素）。
- 确定系统主要角色相关联的用例（Use Case），及每个用例中的主要操作流程，即用例中的必选流和可选流，明确每种角色通过执行用例来履行业务职责时，每种操作流程所用的工作量比例或耗时百分比。
- 确定评测指标和标准，用于评估既定性能目标是否已达成。评测指标通常包括响应时间限度或吞吐量。

最终形成如表 9-1 所示的例子中的系统负载模型。

表 9-1　系统负载模型

<table>
<tr><th>角色</th><th>业务比例</th><th>用例名称</th><th>业务名称</th><th>百分比</th><th>业务描述及简要操作步骤</th><th>监控点</th><th>测试数据描述</th><th>数据要求</th></tr>
<tr><td rowspan="4">注册用户</td><td rowspan="4">20%</td><td rowspan="2">产品信息查询</td><td>丛林冒险活动查询</td><td>33%</td><td>1>.点击主页
2>…</td><td rowspan="2">返回页面名称</td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>登山冒险活动查询</td><td>66%</td><td></td></tr>
<tr><td rowspan="2">产品定购</td><td>产品查询</td><td>60%</td><td rowspan="2"></td><td rowspan="2">交易返回码</td><td rowspan="2">用户信息</td><td rowspan="2">可恢复</td></tr>
<tr><td>产品定购</td><td>40%</td></tr>
<tr><td rowspan="2">过客</td><td rowspan="2">80%</td><td rowspan="2">产品信息查询</td><td>丛林冒险活动查询</td><td>70%</td><td>1>.点击主页
2>…</td><td rowspan="2">返回页面名称</td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>登山冒险活动查询</td><td>30%</td><td></td></tr>
</table>

在 RPT 工具中可以使用性能调度（Schedule）来组织测试场景，完成构建负载模型的任务。调度允许测试员在远程聚合测试，排序测试和运行测试。一个调度可以简单到就

像一个用户在运行一个测试，或者复杂到不同组的成百上千个用户，每一个人在不同的时间运行一个不同的测试。根据所了解到的内容，可以定制这样一个调度：

- 聚合测试来模拟不同用户的行为。
- 设置测试运行的顺序，即顺序地、随机地或加权顺序。
- 每次测试运行时设置时间。
- 以确定的速率运行测试。
- 在远程终端运行一个测试或一组测试。

在建立了一个描述系统行为的调度后，既可以使用正在测试的应用程序的连续构建，又可以使用一个不断增加数量的虚拟用户来运行这个调度。

RPT 调度主要提供了以下测试控件，帮助测试员组织测试脚本，使其满足实际场景，实现灵活的负载模型。

① 用户组：实现不同角色的模拟。用户组让测试员使用多种特性来表达系统上的各种类型的用户，在逻辑顺序下分组测试。对于更现实的用户来说，可以将浏览人员、购买人员、店主或查看订单的用户分组。可以为每一个行为建立一个脚本。在用户组中，可以加入各种测试脚本、随机选择器、循环、延时等完成与角色关联的各种典型业务操作流程的模拟。对用户组可以设置具体业务负载百分比，来模拟不同用户组对被测系统造成的负载比例。

② 随机选择器：增加一个随机选择器，就可以随机地重复一系列的测试，模拟真实用户的不同活动。随机选择器实现用户组内部各种随机业务操作(用例及其事件流)所占不同负载比例的模拟。可以在随机选择器中加入不同的加权块，代表不同的业务操作(用例及其事件流)，通过对其设置权重，完成对其负载比例的模拟。假设一个随机选择器包括两个测试：浏览和下订单。分配“浏览”测试权重 7，“下订单”权重 3。每次执行循环时，“浏览”测试有 70%的机会被选中，“下订单”测试有 30%的机会被选中。

③ 循环：完成用户的重复操作的模拟，例如用户在查询产品时，可能会对不同产品进行多次查询，这时性能测试员可以通过对测试脚本进行参数化和指定脚本的循环次数，来完成对应的负载模拟工作。

④ 延时：用来模拟真实环境中，用户在进行不同业务操作中，可能存在的思考和等待时间。

通过 RPT 中的性能调度(Schedule)，性能测试员就可以将表 9-1 所述的负载模型准确的模拟出来。图 9-11 是一个 RPT 创建场景的例子，其中 Schedule Element Details 提

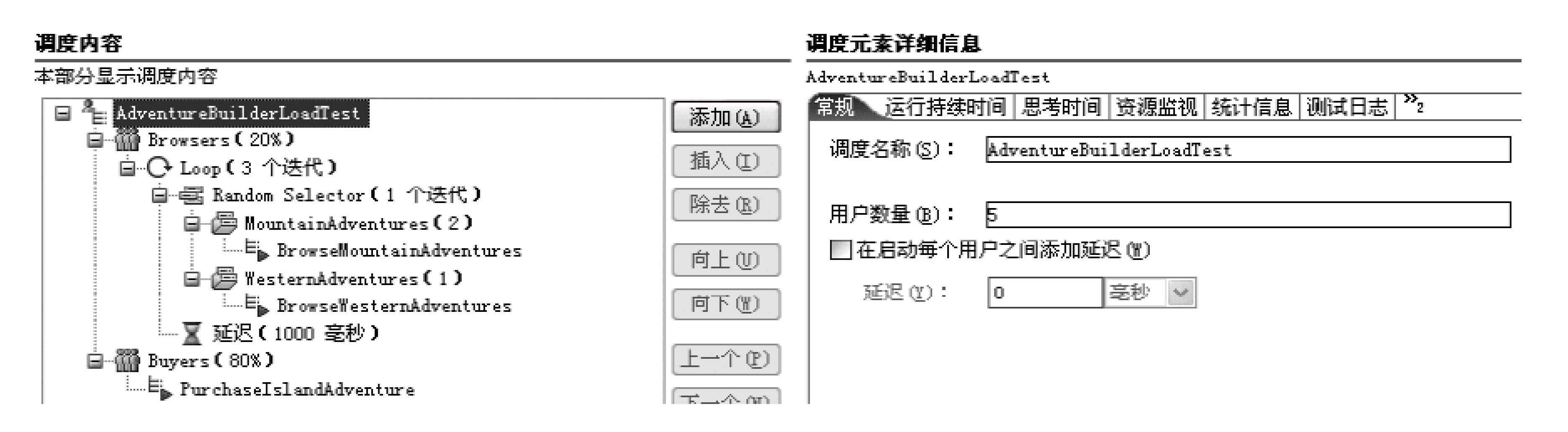


图 9-11　RPT 场景创建

供了对 Schedule 的丰富配置功能。在完成负载模型以后,测试员就可以在“性能调度”的属性中动态随需要指定想要运行的虚拟用户数,进行不同数量的虚拟用户负载情况的性能测试。

RPT 的负载生成器配置也不复杂。首先选中要放到其他 RPT 主机上进行模拟的用户组,然后在其配置界面中选中 Run this group on the following locations,在其中添加远端的 RPT 主机信息。

测试员执行指定的性能调度时,如图 9-12 所示,应该首先选择“运行”,然后在弹出的运行配置管理窗口中,为 Schedule 创建新的配置。在配置的属性中,性能测试员可以指定要执行的 Schedule 执行结果的存放项目和目录,以及是否该配置出现在运行或调试菜单中。

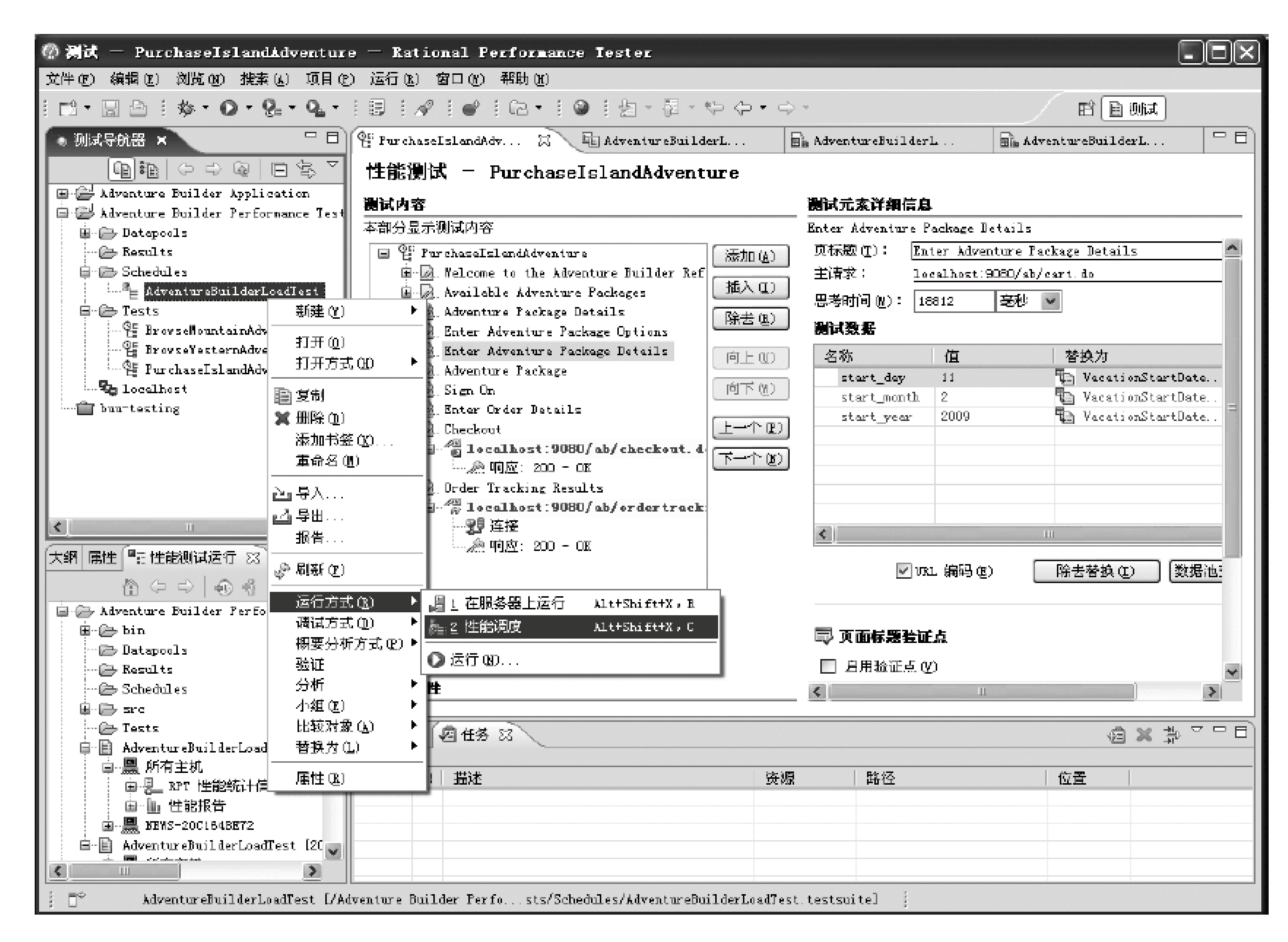


图 9-12 在 RPT 中指定不同数量的虚拟用户执行性能调度

从以上部分可以看到,RPT 为性能测试员提供了通过鼠标单击就可实现的灵活负载模型建立能力,使得负载模型的建立变得异常简单。

9.2.3 性能监控功能

为了更好地进行性能问题分析,找出被测系统的性能问题,性能测试员还需要了解在性能测试的运行过程中,被测系统运行的服务器上的各种系统资源消耗情况。RPT 通过其强大的概要分析(Profiling)能力,可以在系统性能测试运行的同时,完成服务器各种系统资源的监控。

RPT 内部都集成了一些实时监控器,RPT 可以对事务、Web 系统、Web 应用服务器等资源进行实时监控。在自动测试过程中的任何时间里,用户都可使用 RPT 以获知系统

的多种性能指标的当前值和变化趋势。

在一个测试场景中，用户需要将被监控的服务器信息加入到资源监控列表中。图 9-13 中，用户需要在 Schedule 的配置窗中的 Resource Monitoring 标签栏添加需监控的服务器。

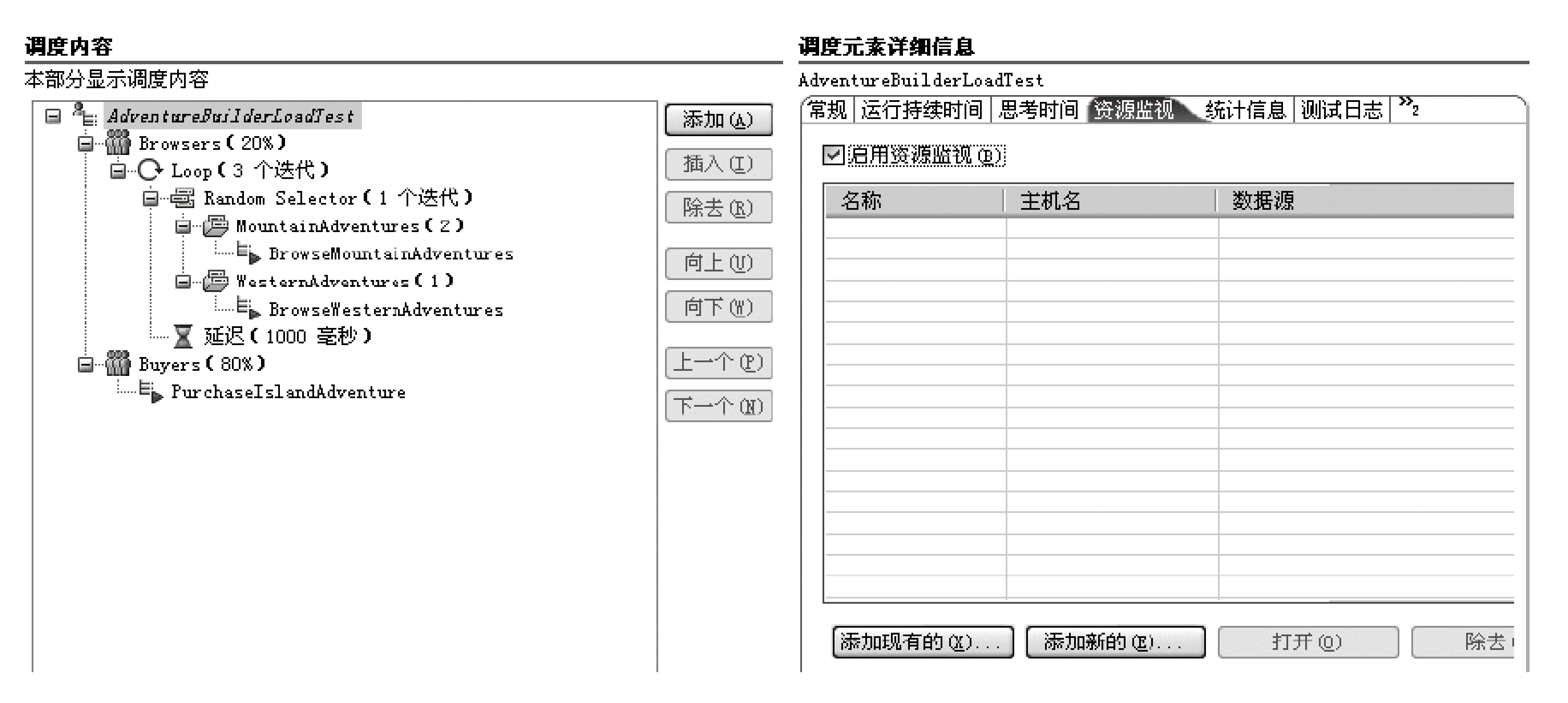


图 9-13　RPT 添加被监控的服务器信息

在“测试透视图”中，性能测试员执行测试时，可以通过选择 Window→Show View→Other→…→Profiling and Logging→Statistical Data 显示“统计数据视图”，实现在性能测试执行的同时，实时监控所关心的各种服务器性能计数器的变化情况。通过将前面分析的各种结果跟被测服务器系统资源消耗情况相关联，性能测试员能够判断是否由系统资源引起了性能问题，从而排除硬件和网络因素对被测系统性能的影响。

9.2.4　测试结果分析

在测试完成后，就可以对测试结果进行分析。这个过程启动了测试和报告的生成程序。测试员将会看到所有的报告，除此之外，还可以为特定的时间范围建立报告。

在进行测试结果分析时，首先性能测试员可以通过总体运行情况报告，对整个测试运行过程便能一目了然。执行测试完成后，只能查看到 6 个报告，只有在运行性能调度时，可以查看到更多的报告。

在查看反馈数据前，需要首先确认已经有了一个比较好的测试运行。采用以下步骤确认测试完成。

① 检查“总体”报告，如果两个或三个柱条的数值都是 100，那么证明运行很健壮。

② 第一个柱条说明页面代码 100％的返回了期望值。在记录过程中，RPT 为所点击的每一页记录下服务器的应对代码。在回放过程中，RPT 将所有虚拟用户所得到的结果与这些数值作比较。任何不匹配的部分将在这里反映出来，如 500-Sever too busy 响应而不是典型的 200-OK 响应。

③ 第二个柱条在页面组件级别上提供了同样的信息。记住页面组件，包括实际的页面 html 以及所有图像和其他页面上的对象。

④ 第三个柱条是在编辑测试脚本时，所设置的验证点的结果摘要。如果是 100％，则

说明所有的验证点都已全部通过。

⑤ 通常来说,希望所看到的柱条都在90%以上。如果没有的话,可以检查“服务器运行状况摘要”和“服务器运行状况详细信息”报告,以确认测试是否有问题。

⑥ 如果需要查看附加的问题信息,可以查看“执行历史记录报告”。

如果确认测试没有问题,那么就可以阅读数据查看其所表现的系统性能。通过查看各个报告来分析数据。

页面吞吐量报告为性能测试员报告了在整个测试运行过程中,虚拟用户的活动情况和页面的吞吐量信息(页面的尝试速率和页面的点击率)。在“页面吞吐量报告”有两个图:页面点击率和用户负载。页面点击率提供了常规服务器响应。如果看到界面上大部分点上页面尝试速率等于页面点击率,那么可以知道,服务器对所有的请求都可以快速地做出应答。如果有比较多的不匹配的地方,说明服务器有时跟上响应很困难。用户负载图显示出在任何给定时间点上的用户数量。首先应该查看报告,以确认用户是否如期望的方式在运行。然后返回这个报告,以确认在这个测试中所经历的瓶颈在给定点处活跃的用户数量。

“页面性能报告”是可用报告最重要的一个,它显示了测试流量中所有页面的平均响应时间。而页面性能报告、响应时间总结报告和响应时间详细报告则为性能测试员提供进一步分析性能问题,定位性能问题提供了必要的信息。页面性能报告显示每个页面的平均响应时间,响应时间总结报告(Response vs. Time Summary)显示所有页面和页面元素的平均响应时间在测试运行过程中的变化情况,性能测试员可以通过它们对被测系统性能问题作出初步估计。响应时间详细报告(Response vs. Time Detail)则详细显示每个页面的响应时间在测试运行过程中的变化情况,为性能测试员提供了更详尽的信息。一般情况下,开始时页面的响应时间要比稳定状态的响应时间慢。

“页面性能报告”是有欺骗性的,平均时间可以掩盖掉突发性的过快或过慢的响应周期,特别是在一个很长周期的测试中。因此,查看“响应与详细时间报告”、检查整个测试流量中的响应时间是极为重要的。如果有部分脉冲,那么需要看一下这些最高点的响应时间,如果没有超过8秒,就不需要特别关注,如果超过的很多,那么需要进行分析。还需要注意有一个典型的模式,就是大多数页面的初始响应时间都要比后续的响应时间慢。这反映出了服务器的高速缓存机制。当服务器对一个页面提供了初始服务,一般情况这个页面会被保存在服务器缓存中。后续的响应可以从这个缓存中得到,使得这个动作会比初始阶段从磁盘获取快得多。在这里可以通过测试起始阶段中从高至低的斜线中查看这个模式。

服务器健康状况的总结报告和详细报告,可以让性能测试员从总体上对服务器运行健康状况一目了然。服务器健康状况总结报告通过显示页面和页面元素总的尝试数、命中数和返回状态码的成功总数,帮助性能测试员方便地了解被测服务器的整体健康状况。而服务器健康状况详细报告则把相关尝试数、命中数和返回状态码的成功总数,进一步细化到每个页面和页面元素,帮助性能测试员准确地了解被测服务器的整体健康状况,定位问题所在。

除以上预定义的各种测试报告外,RPT还为测试员提供了灵活的测试报告的定制能力。通过管理报告功能,性能测试员既可以建立新的报告,也可以编辑各种已经预定义好的报告,修改报告内容或增加报告页面。性能测试员可以根据测试要求,把自己关心的多项指标,显示在一个报告页面。

还有其他报告选项可以查看帮助文档，但是只要运行这些报告选项，就至少需要花费一周的时间。

RPT 提供了测试结果的多种图表以及图表之间的叠加效果，方便用户分析测试结果。RPT 为测试结果提供了直观的图表表现，在默认情况下，用户可以直接看到对测试成功率的总体柱状图、整个测试过程完成的总体信息列表和测试中页面的反应时间曲线图等报告，也可以通过添加其他监控信息的方法，将其他资源的监控图叠加到当前的监控图中，如图 9-14 所示。

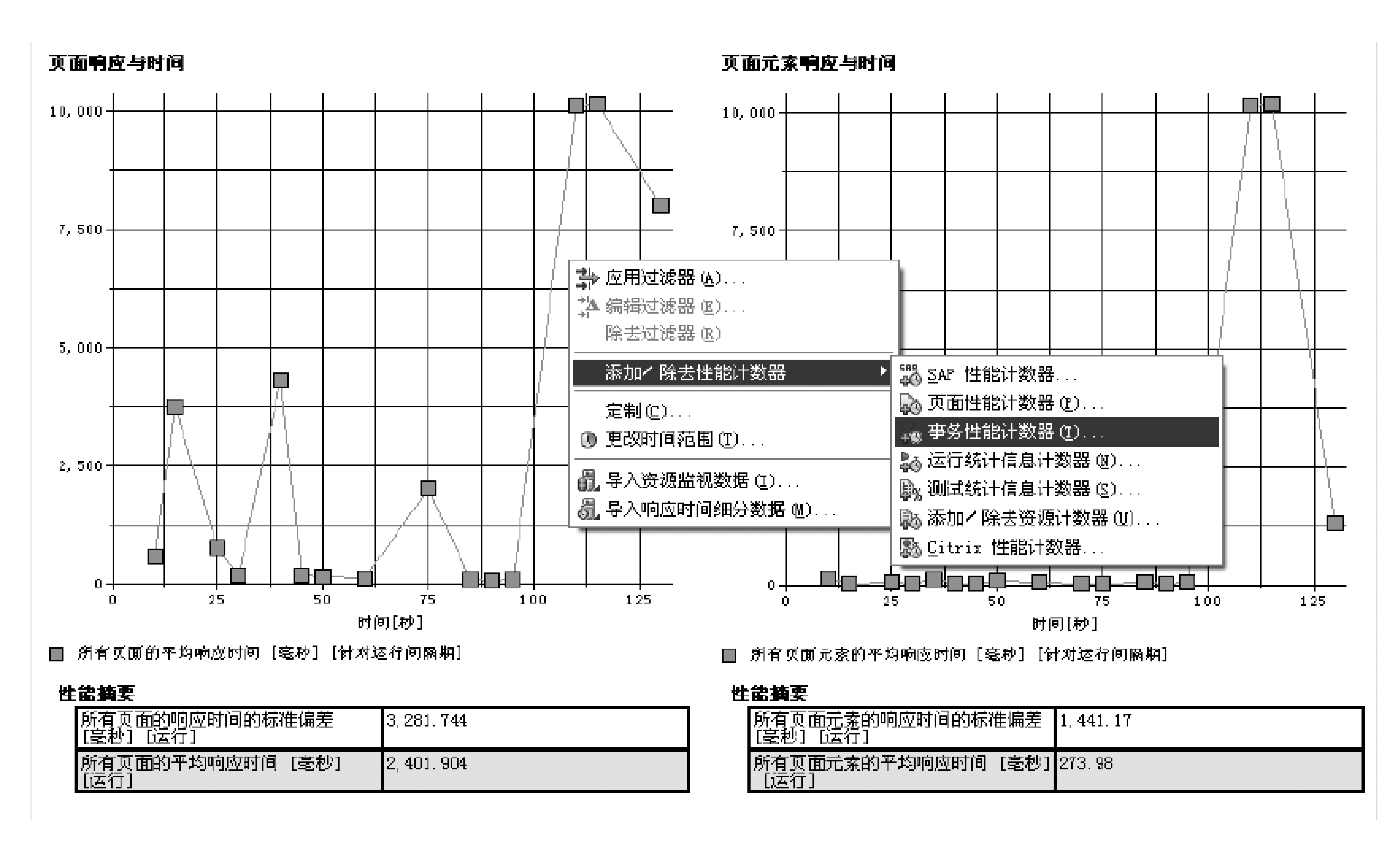


图 9-14　RPT 叠加其他资源监控图

RPT 也可以实现多次测试结果的比较。在 RPT 的 Test Navigator 中选择待比较的测试结果，在其右击的快捷菜单中选择 Compare，打开 Compare Results 窗口，然后将需要做比较的测试结果添加进来，在下一步中选择要显示的报告，点击 Finish 按钮打开比较结果的显示页面。图 9-15 是一个测试结果的比较报告。

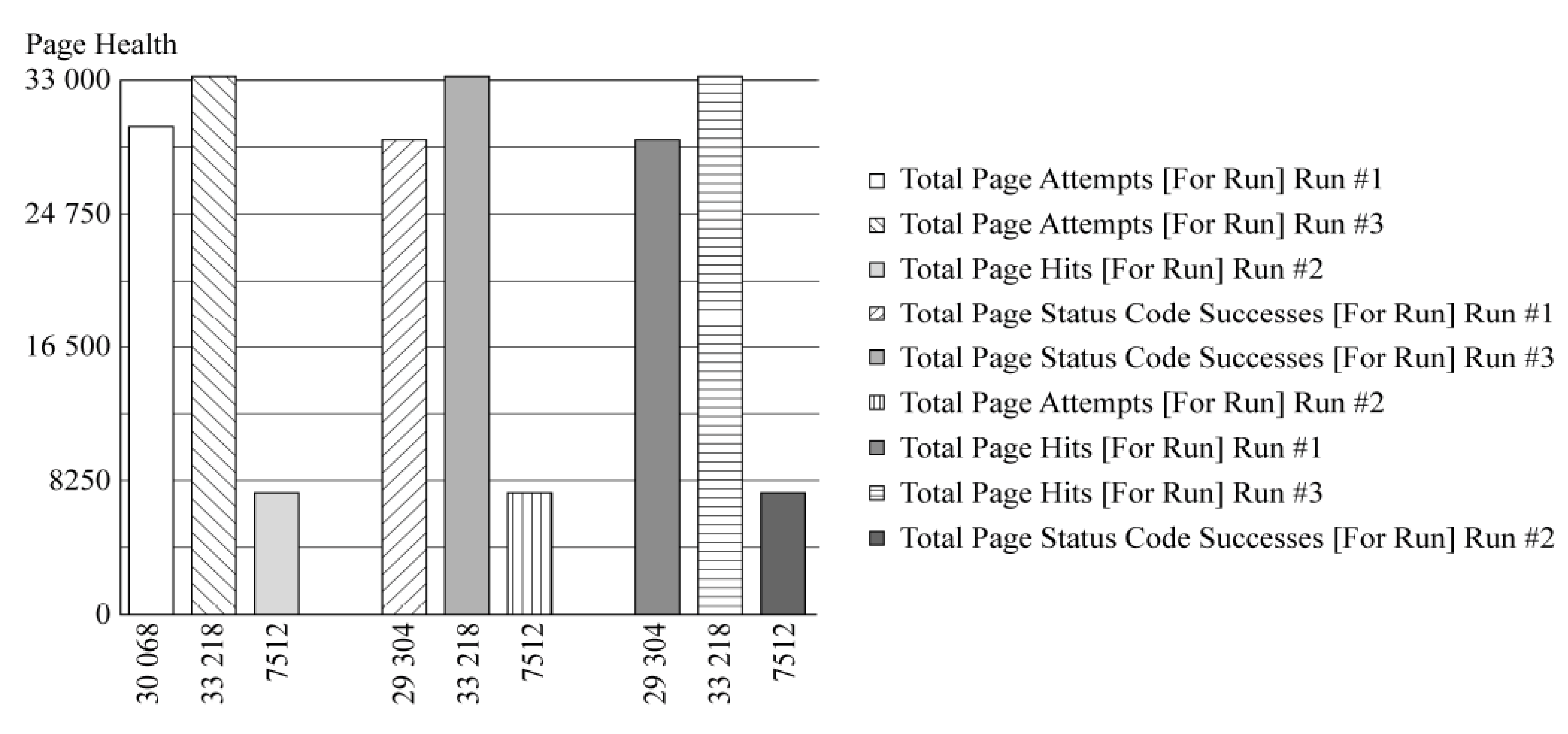


图 9-15　RPT 多次历史测试结果之间的比较报告

RPT 报告定制能力中最有创造性的技术莫过于把性能测试员和用户所关心的各种性能指标以计数器(Counter)的形式记录下来,测试员在生成报告时,可以根据需要从已定义的各种计数器中,选取任何关心的计数器生成报告。这大大增强了性能测试员对测试结果的分析能力和报告能力,这些无疑都会大大改善性能测试员的测试过程体验。

9.2.5 性能调优

所谓性能调优是为了改善系统某些方面的性能而对系统软件或硬件进行的修改。性能调优不是测试人员的职责,性能测试工程师的主要任务是发现并定位性能问题。对于性能测试中发现的问题,通常由性能测试工程师、DBA、系统管理员、开发人员共同来解决。但是对于测试人员,了解调整的相关知识则是十分必要的。

现在通过性能测试,测试员可以检查出开发的应用程序正显示出性能问题的征兆,用户可能会发现该应用程序变得出乎意料地慢。它可能在一段时间之后变为不响应。或它可能没有警告就发生故障,结果导致丢失客户数据。性能问题的原因主要有两类:

① 编码问题:问题出在应用程序本身的逻辑中。编写较差的应用程序代码,可能导致应用程序执行多余的或者不必要的工作。例如,算法效率低或伸缩性不好,不必要地频繁调用和远程调用,SQL 查询效率低等,都会导致应用系统性能问题。

② 配置问题:问题与应用程序无关,而是由外部因素引起的。这些因素包括硬件问题(如内存不足或处理能力不足)、网络问题(如等待时间长、吞吐量低和连接问题)以及其他相关软件问题(如数据库调整不当)。

如何从复杂的测试报告来进行分析问题所在呢?一个普遍遵循的原则是“由外而内,由表及里,层层深入”,对于一个应用系统,性能开始出现下降的最直接表象就是系统的响应时间变长。于是系统响应时间成为分析性能的起点。首先应该从原始测试数据中查看系统响应时间,判断它是否满足用户性能的期望。如果不能满足,则说明系统的性能出现了问题。发现系统存在问题后,就要判断系统在哪个环节出现了瓶颈。

现在的 IT 系统架构极其复杂,任何一个环节出现瓶颈,都会导致系统出现性能问题。要准确地判断瓶颈在什么地方,的确是一个棘手的问题。不过,任何复杂的系统都分为网络和服务器两部分。因此要考察的第二个问题就是系统的瓶颈出现在网络环节还是服务器环节。

如图 9-16 所示,用户从客户端发起的请求数据包经过网络,传递到应用服务器,最后到达数据库服务器,服务器处理完毕后按原路返回到客户端。在这个处理过程中,可以把整个时间分为两段:一段是 Tn,即网络的响应时间;一段是 Ts,即服务器的响应时间,包括应用服务器和数据库服务器的响应时间。对比 Tn 和 Ts,就很容易知道系统在哪些环节的响应时间比例较大。

只要判断出系统的瓶颈是出现在网络或是服务器段,就可以层层推进对相应环节的组件响应时间进行深入分析,直到最后找到造成性能问题的根本原因。

在分析出问题发生的原因后,测试人员和系统调整人员首先要确定调整目标,然后设计解决方案。确定调整目标的主要作用是明确何时停止系统调整,否则工作将永无尽头。

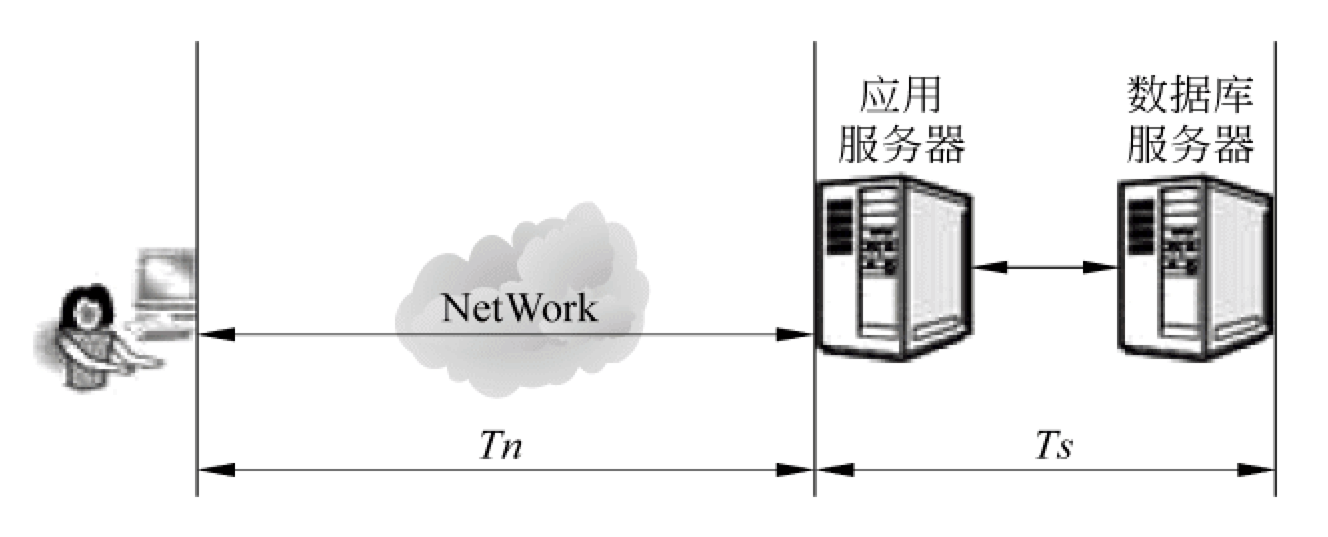


图 9-16 客户交易分解图

每个系统都有不同的特点，因此调整目标可能各有不同。例如系统的调整目标有提高系统吞吐量、缩短响应时间、更好地支持并发。

设计解决方案的主要依据就是这些调整目标。有了明确的方案和目标，就可以进行后面的工作了。实施解决方案后，就要对方案进行测试。可以使用以前的测试用例来进行测试，验证系统是否解决了性能问题。测试解决方案尽量要在仿真环境下进行，因为在生产环境下可能会带来破坏，除非充分估计了测试的风险，并且准备了万全的补救方案。

性能调优的最后一步是分析调优结果，如果问题没有得到解决，则要重复前面的工作。在测试系统调优方案过程中，要经常分析所做的工作。如果没能准确定位问题或调整方案不正确，可能会达不到预期目标。要尽早发现这些错误，以使工作早些回到正确的轨道上来。达到预期目标后，调优工作基本就可以结束了。

IBM Rational 的性能测试解决方案除了 RPT 提供的各种性能测试功能以外，还包括最新发布的免费性能优化工具包(IBM Performance Optimization Toolkits，IPOT)，它有助于在分布式应用程序中，找出并修正与编码相关的性能问题。与性能测试工具 RPT 配合使用，帮助性能测试员准确定位基于 J2EE 的 Web 应用系统性能问题，分析问题根源，解决问题。利用性能优化工具包可以从生产或开发环境运行的应用程序中，收集应用性能数据。帮助性能测试员解决以下多种不同类型的性能问题：

① 性能问题：执行操作的时间比预期的长。

② 内存泄漏：应用程序在内存使用方面处理不当，导致正常操作期间发生内存不足的错误。

③ 应用程序故障：应用程序发生故障，表现为有警告或无警告地突然终止(崩溃)，或进入不响应状态(挂起)。

9.2.6 实用技巧

1. 调整日志采样频率和粒度以适应不同的测试场景

RPT 的 Schedule 提供了测试数据的采样频率和采样粒度的配置，以适应不同的测试场景。在做长时间测试时，在 Schedule 的配置面板中的 Statistics 标签，通过适当增加日志采样间隔时间、日志级别和采样用户量，降低 RPT 的压力，避免因采样数据量过大，使内存耗尽，导致 RPT 无法响应，如图 9-17 所示。

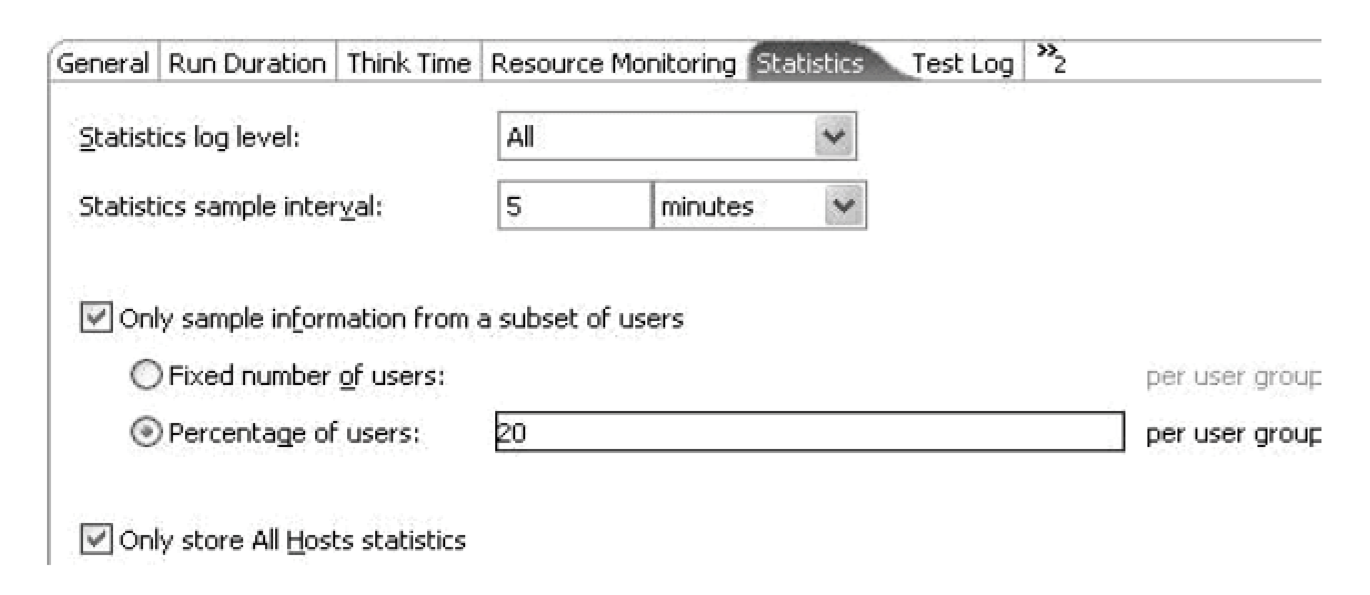


图 9-17　Statistics 配置

在测试过程中，RPT 会记录测试中的请求数据和响应数据，以便在测试之后查看测试过程中的数据和错误信息。对于长时间测试，会产生大量的请求和响应数据，这些大量的日志信息会让 RPT 不堪重负。在 Schedule 配置部分的 Test Log 标签中提供了日志记录的级别设置，对于长时间的测试，推荐使用图 9-18 所示的配置，记录错误和警告信息的级别为 All，而对于其他信息则只记录 Primary Test Actions 即可。

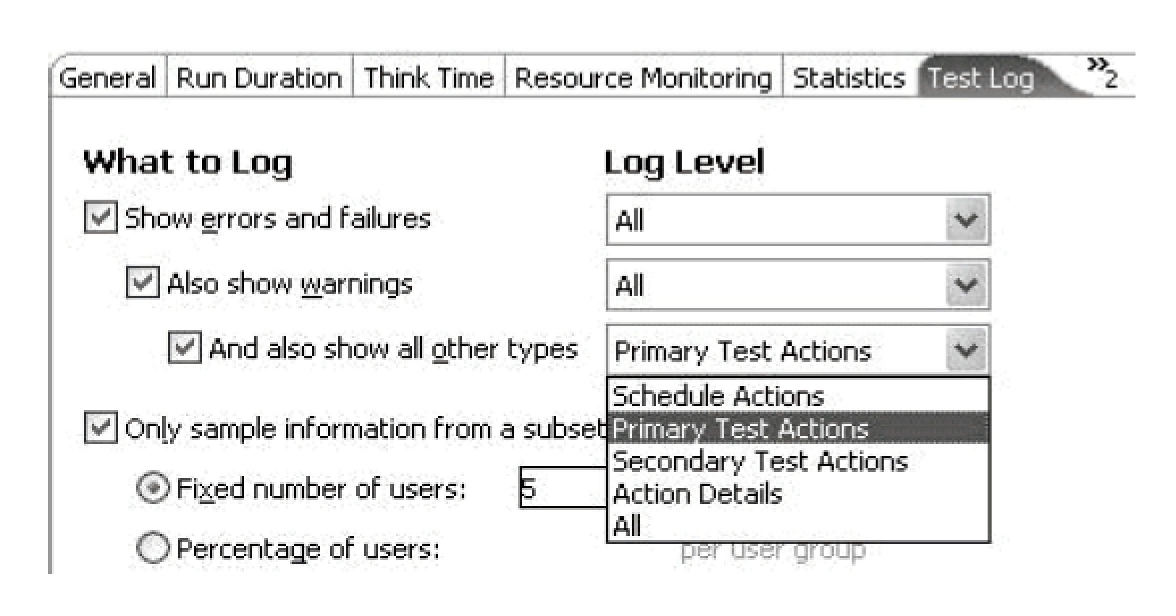


图 9-18　LOG 配置

2. 通过 Custom Code 实现多条测试数据的随机读取

在 RPT 中通常采用数据池的形式作为少量测试数据，例如模拟多个用户登录所用的多个用户名密码信息的输入。不过 DataPool 的读取方式为顺序读取。如果对于输入数据需要随机读取，则可以通过 Custom Code 来实现。其实现方式可以是将测试数据存为 *.cvs 等格式文件，然后通过文件操作，并根据随机数从文件中读取内容作为测试输入数据。

3. 使用超大测试数据集文件

对于在测试过程中存在少量测试数据分次作为测试输入数据时，例如模拟多个用户登录所用的多个用户名密码信息，通常采用数据池的形式。但是在测试数据量较大时，该方法就不再适用，因为对于数据池中的数据，测试开始时就被全部加载到内存，如果测试数据量过大，这种方式会造成大量的内存浪费。

这种情况可以通过 RPT 提供的扩展功能自定义代码来定制代码，达到在测试过程中需要的时候再去加载所需的内容，也可以采用文件操作的方式来实现。

9.3 小　　结

开展性能测试工作，仅有 RPT 软件是远远不够的，深入地理解性能测试理念是做好性能测试工作的前提。因此在开始学习 RPT 软件前，应该先搞清楚“性能测试是怎么一回事”，并学会如何设计与组织性能测试。

本章首先从性能测试的基本概念入手，介绍了性能测试的相关知识和如何做性能测试工作，还介绍了性能测试调整方面的一些知识，以掌握性能测试调优的“度”。

掌握了性能测试的基础知识后，将逐步踏入 RPT 软件的性能测试世界。RPT 是一种帮助开发团队在配置前确认可测量性和可靠性的性能测试创建、执行和分析工具。本章概要介绍了 RFT V 7.0 下运行负载测试的基础知识，还从多方面进行详细的分析，并根据实践经验总结了一些 RPT 的实用技巧，以更加合理地开展性能测试。

习题与思考

1. 什么是性能测试？
2. 性能测试的目标是什么？
3. 如何解决性能问题？
4. 性能测试方法常用的有哪些？简述各种方法主要用途。
5. 性能测试员进行系统性能测试的过程中主要面临哪些挑战？
6. 对 IBM Rational Performance Tester 进行简单介绍。
7. RPT 对系统性能进行分析的过程包括哪几个步骤？
8. 描述创建数据池的步骤。
9. 描述测试结果分析的步骤。
10. 什么是性能调优？性能问题的主要原因的是什么？

第5部分

验收测试

如果猴子大军狂敲打字机，它们就可以写出大英博物馆中的所有藏书。

——英国物理学家 Arthur Eddington

Arthur Eddington 的话源于一个想法，如果让一百万只猴子在一百万只键盘上敲一百万年，它们最终可能写出莎士比亚话剧《好奇的乔治》等巨著。当软件公开发布时，成千上万的人会乱敲乱试，也可能无意间碰巧发现其他方式会漏掉的软件缺陷，这也是验收测试的一种方式。

RUP 认为验收测试是部署软件之前的最后一项测试操作。验收测试的目的是确保软件准备就绪，并且可以供最终用户用于执行软件的既定功能和任务。验收测试既可以是非正式的测试，也可以是有计划、有系统的测试。有时，验收测试可能长达数周甚至数月。一个软件产品可能拥有众多用户，不可能由每个用户验收，此时多采用称为 α、β 测试的过程，以期发现那些似乎只有最终用户才能发现的问题。

易用性和无障碍测试是软件验收测试的重要内容。本部分包括第 10 章和第 11 章，对软件的易用性测试和无障碍测试的技术、方法作了重点介绍，其中无障碍测试是本书的一大特色，在我国软件测试领域可能是第一次详细介绍。

第10章 易用性测试

易用性笑话

第一次走到新的办公室门口，用劲去推那扇巨大玻璃门，同时脚步并没有放慢下来。

结果“哐”的一声巨响，我差点整个身子都撞了上去，鼻子已经贴到了玻璃上。原来，这扇门只能拉，而不能推。

这是好多人都可能会遇到的事情：只能拉的门去推，只能推的门去拉，或者直接一头撞进那种左右滑动的移动门上去……这个时候常会自责“真不小心”！其实，傻的不是我们，而是门的设计师。门的把手、遥控器，以及任何我们周围的东西，都是用户界面。让人会犯错误的设计，是易用性出了问题。易用性差的门，就像给我们设计的圈套，等着我们掉进去。

这是易用性专家Donald A. Norman在他的书《The Design of Everyday Things》举的关于门的例子，这充分说明了易用性的重要性，特别是随着WWW技术的发展和电子政务的兴起，易用性显得更为重要。

软件是高新技术，人们对软件的认识通常是从技术上来考虑，似乎技术越先进，水平越高，系统就越好。这里指的人们，不仅指设计人员和管理人员，而且包括普通用户。因此提出软件的易用性问题，不仅是设计人员思想上的变革，也是普通人认识上的进化。

易用性的测试和评估涉及心理学、艺术、软件工程等多学科领域，现在国外已经形成一个新的专业——易用性工程(Usability Engineering)，发展并形成了一整套方法和技术，来进行易用性的测试和评估。国内相当一部分软件企业还没有认识到易用性的价值，或者说认识到了易用性的价值，却还缺乏紧迫感，没有真正实施开展起来。

学习本章读者需要了解优秀UI的要素，思考如何根据这些要素来开展测试。本章还讨论了Web的易用性，介绍了易用性的标准和规范，并介绍了根据易用性质量指标体系对微软产品进行易用性测试、评估，以及如何评测网站和Web系统的易用性，保证Web系统具有良好的易用性。

10.1 易用性测试基础

10.1.1 易用性的定义

也许大家有过分别使用功能类似的不同软件的经历，通常情况下，将会保留“感觉更好用”的软件，并一直使用下去。任何比较都存在一个标准，而用户使用何种标准对功能类似的软件做出“好用”和“不好用”的主观判断呢？这个标准就是软件的易用性[①]。

那么到底什么是易用性呢？对于这个问题，目前国际上还没有统一的说法，但可以通过该领域和相关领域研究人员的解释来理解易用性。

20 世纪 90 年代初，针对计算机界面的使用日益频繁这一现象，曾被国际互联网杂志称为“易用性之王”的 Jakob Nielsen，把对产品的使用分为可用和易用。而易用性又分为以下 5 个属性。

① 易学性：指用户很容易就能掌握如何使用产品。

② 高效性：使用有效，能很快实现预期目标。

③ 易记忆：隔一段时间（几天或几个月）后，再次使用，不需要学习就能自如操作。

④ 少犯错：操作过程不易犯错，即使犯错也能及时发现并纠正。

⑤ 满意度：用户对产品表示满意并乐于使用它。

国际标准化组织在 ISO9241—11 中，对易用性的定义为：“在一定程度上，一个产品能在特定的领域里由特定的用户使用，并使他们能有效地、高效地、满意地实现特定目标的性能。”该定义将易用性分为有效性、高效性和满意度三个属性。而用户、用户目标和使用环境也是影响易用性的重要因素。可以发现它们之间存在一定的联系。所有对产品的易用性属性的描述最后都可归结到人的生理接受性和心理接受性上，由此可见对用户的关注和了解是实现产品易用性的保证。

10.1.2 优秀用户界面的要素

软件也是商品，因而也同样具有易用性的问题。那么软件的易用性表现在哪里呢？在当今社会，计算机存在于人们工作和生活的方方面面。与早期的计算机用户不同，今天的用户绝大多数没有受过专业的培训，对计算机内部如何处理并不了解也不关心，只是想利用计算机来达到自己的目的。他们更愿意用计算机来工作，而不是去学习和掌握那些复杂的工具。人们期望不用经过专门的培训，就可以使用计算机来解决更多的问题。就

① “易用性”译自英文 Usability，可以直译为“可用性”，台湾有学者翻译为“优使性”，由于“可用性”很容易被误解为对产品是否可用的评估，而“优使性”不符合大陆的阅读习惯，本书考虑了其在软件工程中的主要应用和中文专有名词的编写习惯，认为易用性更能表达出其含义和特性。本书以“易用性”来表示 Usability，读者阅读其他文献时，请以文献表述为准。

像人们可以直接驾车去商店购物，而不用了解发动机是如何工作的一样。人们是通过使用各种软件来与计算机打交道的，而软件产品一般是用户界面呈现给用户的，所以用户常常是通过软件产品的用户界面来评价一个软件产品的。也就是说，软件的易用性主要表现在用户界面上。

用户界面(UI)指用于与软件程序交互的方式。所有软件都有几种用户界面。纯粹主义者可能会说这不对，像汽车中控制发动机空燃比的软件就没有用户界面。事实上，它只是没有传统的用户界面，但是拉风门增大气压并从排气管听到噼啪响声其实就是用户界面。虽然这些用户界面各不相同，但是从技术上讲，它们与计算机进行同样的交互——提供输入和接受输出。

下面是优秀用户界面常见的 7 个要素。无论用户界面是电子表还是 Mac 操作系统界面等，它们都适用。

1. 符合标准和规范

最重要的用户界面要素是软件符合现行标准和规范或有真正站得住脚的理由。如果软件在 Mac 或 Windows 等现有平台上运行，标准是已经确立的。Apple 的标准在 Addison-Wesley 出版的《Macintosh Human Interface Guidelines》一书中定义，而 Microsoft 的标准在 Microsoft Press 出版的《Microsoft Windows User Experience》一书中定义。两本书都详细说明了该平台上运行的软件，对用户应该有什么样的外观和感觉。每一个细节都有定义何时使用复选框而不是单选按钮，即何时两种选择状态是完全相反的或不清楚，何时使用提示信息、警告信息或严重警告。

注意：如果测试在某个平台上运行的软件，就需要把该平台的标准和规范作为产品说明书的补充内容，像产品说明书一样，根据它建立测试案例。

这些标准和规范由软件易用性专家开发。它们是经过大量正规测试、使用、尝试和纠错而设计出的方便用户的规则。也并非要完全遵守准则，有时开发小组可能想对标准和规范有所提高。平台也可能没有标准，也许测试的软件就是平台本身。在这种情况下，设计小组可能成为软件易用性标准的创立者。

2. 直观性

当测试用户界面时，应考虑以下问题以及如何衡量自己软件的直观程度。

- 用户界面是否洁净、不唐突、不拥挤？用户界面不应该为用户制造障碍；所需功能或期待的响应应该明显，并在预期的地方出现。
- 用户界面的组织和布局合理吗？是否允许用户轻松地从一个功能转到另一个功能？下一步做什么明显吗？任何时刻都可以决定放弃或者退回、退出吗？输入得到承认了吗？菜单或者窗口是否深藏不露？
- 有多余功能吗？软件整体功能亦或局部功能是否做得太多？是否有太多特性把工作复杂化了？是否感到信息太庞杂？
- 如果其他所有努力失败，帮助系统能帮忙吗？

3. 一致性

应用软件的界面设计注重一致性。设计者要密切注意在相同应用领域中最流行的软件界面，必须尊重用户使用这些软件的习惯。例如商业软件习惯于设置 F1 键为帮助热键，如果某个设计者别出心裁地让 F1 键成为程序终止的热键，那么在用户渴望得到帮助而伸手按 F1 键的一刹那，他的工作就此结束。相信这个用户会“一朝被蛇咬，十年怕井绳”。

目前流行的软件开发工具如 Visual C++、Visual Basic、Delphi、C++ Builder、Power Builder 等，都能够快速地开发出非常相似的图形用户界面。在 Internet/Intranet 领域，浏览器几乎成了唯一的客户机程序，因为用户希望用完全一致的软件来完成千变万化的应用任务。

如果软件或者平台有一个标准，就要遵守它。如果没有，就要注意软件的特性，确保相似操作以相似的方式进行。在审查产品时应检查以下选项：

- 快速键和菜单选项。在语言信箱系统中，按 0 键，而不按其他数字，几乎总是代表接通某人的“拨出”按钮。在 Windows 中，按 F1 键总是得到帮助信息。
- 术语和命令。整个软件使用同样的术语吗？特性命名一致吗？例如 Find 是否一直叫 Find，而不是有时叫 Search？
- 听众。软件是否一直面向同一听众级别？带有花哨用户界面的趣味贺卡程序就不应该显示泄露技术机密的错误提示信息，因为这两个不是面向同一听众。
- 按钮位置和等价按键。大家是否注意到对话框有 OK 按钮和 Cancel 按钮时，OK 按钮总是在上方或左方，而 Cancel 按钮总是在下方或右方？同样的原因，Cancel 按钮的等价按键通常是 Esc 键，而选中按钮的等价按键通常是 Enter 键。

4. 灵活性

根据用户喜好提供选择，但不要太多，足以允许他们选择做什么和怎样做。例如 Windows 计算器程序有两种视图：标准型和科学型。用户可以决定用哪个来完成计算。当然灵活性越强，测试的复杂性就增大。在计算器例子中，两个视图相对一个视图就需要做更多的测试。

- 状态跳转。灵活的软件实现同一任务有多种选择和方式。结果是增加了通向软件各种状态的途径。状态转换图将变得更加复杂，软件测试员需要花费更多时间决定测试哪些相互连接的路径。
- 状态终止和跳过。当软件具有用户非常熟悉的超级用户模式时，显然能够跳过众多提示或窗口直接到达想去的地方。能够直接拨到公司电话分机的语言信箱就是一个例子。如果测试具有这种功能的软件，就需要保证在跳过所有中间状态或提前终止时正确设置状态变量。
- 数据输入和输出。用户希望有多种方法输入数据和查看结果。为了在写字板文档中插入文字，可以用键盘输入文字，粘贴文字，从多种文件格式读入文字，文字作为对象插入，或用鼠标从其他程序拖动文字。Microsoft Excel 电子表格程序，

允许用户以 14 种标准和 20 种自定义图形的形式查看数据。测试进出软件的各种方式，将极大增加工作量，使等价分配难以抉择。

5. 舒适型

软件应该用起来舒适，而不应该为用户工作制造障碍和困难。软件舒适性是讲究感觉的，研究人员想找出软件舒适的正确公式，这是难以实现的理论，但是可以找到如何鉴别软件舒适性的一些看法：

- 恰当。软件外观和感觉应该与所做的工作和使用者相符。如果界面不适合于软件的功能，那么界面将毫无用处，界面美的内涵就无从谈起。所以界面的合适性是界面美的首要因素，它提醒设计者不要片面追求外观漂亮而导致失真或华而不实。界面的恰当性既提倡外秀内惠，又强调恰如其分。恰当性差的界面无疑会混淆软件意图，致使用户产生误解。即使它不损害软件功能与性能，也会使用户产生不应有的情绪波动。例如一些软件开发者喜欢为其作品加一段动画演示，以便吸引更多用户的关注。这本是无可非议的，问题在于这演示是否合情合理。如果运行一个程序，它首先表演一套复杂的动画，在后台演奏雄壮的进行曲，电闪雷鸣之后出来的却是一个普通的文本编辑器。整个过程让用户置身于云里雾里，而结果却让用户感到惊愕而不是惊喜。不恰当的界面只会给软件带来厄运。
- 错误处理。程序应该在用户执行严重错误的操作之前提出警告，并且允许用户恢复由于错误操作导致丢失的数据。现在大家认为 Undo/Redo 特性是当然的，但是原来的软件根本没有这些特性。
- 性能快慢适中。有些程序的错误提示信息一闪而过，无法看清。如果显示缓慢，应该向用户反馈操作持续时间，并且显示它正在工作，没有停滞。

6. 正确性

判断舒适性的界定不那么明确，很大程度凭个人感觉。而测试正确性，就是测试用户界面是否做了该做的事。此类正确性问题一般很明显，在测试产品说明书时就可以发现问题。然而，以下情况要特别注意：

- 市场定位偏差。有没有多余的或遗漏的功能？或者某些功能执行了与市场宣传资料不符的操作？注意不是拿软件与说明书比较，而是与销售材料比较。这两者通常不一样。
- 语言和拼写。程序员知道怎样只用计算机语言的关键字拼出句子，常常制造一些非常有趣的用户信息，常人却看不明白。下面是来自一个流行电子商务网站的订单确认信息，“下列信息如有不符，请立即与我们联系，以确保及时得到预订的产品。”这个信息，普通用户看了可能有些糊涂。
- 不规范的多媒体。多媒体是软件用户界面包含的所有图标、图像、声音和视频。图标应该同样大，并且具有相同的调色板。声音应该都有相同的格式和采样率。正确的媒体从用户界面选择时应该显示出来。
- WYSIWYG(所见即所得)。保证用户界面所说的就是实际得到的。当单击 Save

按钮时，屏幕上的文档与存入磁盘的完全一样吗？从磁盘读出时，与原文档相同吗？

7. 实用性

优秀用户界面的最后一个要素是实用性。在审查产品说明书时，想一想看到的特性对软件是否具有实际价值。它们有助于用户执行软件设计的功能吗？如果认为它们没必要，就要研究一下，找出它们存在于软件中的原因。

10.1.3 易用性原理

易用性原理可归结为三点：易见、映射和反馈。

① 易见(Visibility)。单凭观察，用户就应知道设备的状态，该设备供选择可以采取的行动。关于门的例子，就是易见性出了问题。通过观察它，没有办法判断应该做哪些操作，是推还是拉？有很多门上贴着“推”或“拉”，其实当一项设计需要用标签的时候，就说明设计本身已经失败了。

② 映射(Mapping)。知识主要分布在两个地方。

- 每个人的脑子里。这些知识对高效地使用一样东西非常有用。
- 环境中。这些知识当第一次遇到时非常有用。

对于大家日常使用的饮水机的两个水龙头，大多数人不加思考就知道红的是热水，蓝的是冷水。这就是产品的设计和我们脑子里已经形成的火是红的，水是蓝的映射。这简单的映射，让大家对一个产品的接受度大大增加。微软的 DOS，就第一次采取了 DIR(目录)作为目录的结构，而不用 ls 这样 UNIX 的文件名，而且第一次使用了形象的 C 盘、D 盘。到了 Windows 时代，更是把目录干脆变成了文件夹的样子，文件变成了一张张写着字的纸。而 Windows 95 更加开创性地第一次引入了桌面的概念，把我的电脑变成桌面的一部分。这种种变化，都是把产品映射到了大家熟识的生活中的概念。笔者曾经用过红颜色代表冷水，蓝颜色代表热水的饮水机，就算有再多的标签告诉那个是热水，可能还是会搞错。

③ 反馈(Feedback)。反馈对于产品的易用性至关重要。通过不断的、迅速的反馈，让用户知道自己的操作结果。例如，Windows 在应用中的那个“沙漏”的鼠标，就是让用户知道需要等待。产品中“正在……请稍后”就好于死机那样一动不动。关于反馈，好多投影仪的易用性设计存在问题。来看一个典型的场景：拿起投影仪遥控器，按开的按钮。等了 3 秒钟，开始不耐烦，因为投影仪虽然已经开动，但是反应慢，需要一些时间预热。我们会以为没有反应，再按一下。这回更耐心一点了，但是投影仪已经被关掉了。15 秒钟以后，接着再按那个开关键，直到愤怒地离开……没有反馈，会让初次使用的用户不知所措。例如 Windows 的登录框，输入密码时，因为反馈被星号屏蔽了，多少次按了大写键以后没法输入密码就是一个易用性问题。好在从 Windows XP 以后，用一个“大写键打开”的提示框作为反馈，就好多了。

此外，易用性是针对不同人的，设计者无法准确知道该产品是否对某些人同样易用。

10.1.4　易用性要点

开展易用性测试的时候，主要注意以下三个要点：

① 运用已有产品的功能造型，创造新的产品。由于人们积累了一定的使用经验，例如门把手的语意指示在哪里按压，在哪里推拉等，所以可以轻易地理解并使用。

② 开发新的共用的功能元素。在使用者能够使用的基础上，创造新的易用性语意，设计不是墨守成规，照搬照套，它是一个不断发展进步的过程。例如，最早期的计算机机箱，它的主板接口位置并没有明确的标示，但随着时代的变化与需求，设计师开始从用户的认知出发，所以后来有了彩色标注并附有中文注明，普通用户也能方便地进行安装。

③ 注重使用者的心理认知度，加强产品易用性的情感因素。易用性是一种语言，它不是单纯建立在高新技术基础上的，情感化因素在设计中日益显示出它的重要性，并在一定程度上对使用者造成积极或消极的心理影响。所以易用性设计应该以此为出发点，而不仅是为了体现一个时代的技术特征。

10.1.5　易用性测试原则

根据给软件易用性所下的定义，一个软件易用性的测试和评估应该遵循以下原则：

① 最具有权威性的易用性测试应由产品用户来完成。

最具有权威性的易用性测试和评估不应该是专业技术人员，而应该是产品的用户。因为无论这些专业技术人员的水平有多高，无论他们使用的方法和技术有多先进，最后起决定作用的还是用户对产品的满意程度。因此，对软件易用性的测试和评估，主要应由用户来完成。

对一些用户而言，“测试”有负面的含义。要努力确保他们不认为测试是针对他们。要让他们明白，他们正在帮助测试原型或网站。事实上，可以不使用“测试”这个术语。相反，是邀请参加者提供帮助，勇于尝试原型。

当用户难以完成任务时，应该改变产品，而不是改变用户。同时还应该思考该产品能在多大程度上符合那些典型用户的目标，而不是关注用户在这个任务的过程中做得多好。

② 软件的易用性测试和评估是一个过程。

这个过程早在产品的初样阶段就开始了。因此一个软件在设计时反复征求用户意见的过程应与易用性测试和评估过程结合起来进行。当然，在设计阶段反复征求意见的过程是后来易用性测试的基础，不能取代真正的易用性测试。但是如果没有设计阶段反复征求意见的过程，仅靠用户最后对产品的一两次评估，是不能全面反映出软件的易用性的。

③ 软件的易用性测试必须是在用户的实际工作任务和操作环境下进行。

易用性测试和评估不能靠发几张调查表，让用户填写完后，经过简单的统计分析就下结论。易用性测试必须是用户在实际操作以后，根据其完成任务的结果，进行客观的分析和评估。

一些设计人员认为，如果他们的设计能迎合用户的喜好，用户在该产品上就会有良好的表现。但证据并不支持这一点。事实上，用户的表现以及他们对产品的偏好并非一一对应。一项研究发现，就浏览网页方面而言，约有70%用户同意表现和喜好有联系。也就是说，他们在喜爱的网站上表现良好，在不喜欢的网站上表现欠佳。然而，还有30%的用户认为，用户的表现以及他们对产品的偏好并非一一对应。他们在不喜爱的网站上可能表现良好，在喜欢的网站上也可能表现不佳。对此用户有多种解释，他们可能担心给一个较低的评价会伤害网站设计者，也就是伤害感情。或者说他们并没有完成任务，却自认为任务完成了，他们并没有意识到问题所在。所以易用性一般依靠用户的表现，而不是他们的偏好。

④ 要选择有广泛代表性的用户。

因为对软件易用性的一条重要要求就是系统应该适合绝大多数人使用，并让绝大多数人都感到满意。因此参加测试的人必须具有代表性，应能代表最广大的用户。

制造任何产品，包括大部分网站和软件，需要考虑许多不同的用户的工作方式、体验、问题以及需要。需要认真考虑假定用户、使用场景以及易用性测试的结果，试图找出针对不同客户需求的理想解决方法。找不到最好的解决方法，用户就不能够顺畅地完成任务。有证据表明，即使用户延长使用时间，在一个不太完美的产品界面完成任务，也远不及在一个更好的产品界面带来的成功感。

所以一般最好优先开发那些能使最多用户完成任务的网站或软件。有研究表明，产品推出后，由于易用性的问题造成客户的困惑，用于客户的技术支持费用远远高于开发时对产品修正所付出的花费。

10.1.6 易用性测试与软件测试的区别

易用性测试是QA的测试的一个组成部分，但易用性测试和软件测试还是有很大区别的，至少不应该属于软件测试范畴。易用性测试和软件测试有各自的定义和适用的范畴。

软件测试的定义(IEEE 1983)：“使用人工和自动手段来运行或测试某个系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别。”

以上可以看出软件测试和易用性测试的区别。软件测试主要是对软件本身质量的评估和控制，侧重点是程序、技术和逻辑，主要是通过寻找Bug的方法来进行的。易用性测试是从用户的角度，模拟用户的使用行为、心理和环境来对软件进行评估，侧重点是能否满足用户的易用性需求和良好的用户体验。

一般来说易用性工程师应该归属于用户体验部门，当然这两者之间还是有重合的地方。但为了使产品有最好的用户体验，需要软件测试和易用性测试一起进行。

10.1.7 易用性与情感的关系

产品的外观或功能带来的乐趣可以让人们产生正面情绪，激发创造力，并且提高对小

困难和小挫折的耐受性，就算产品的设计中存在小问题也会被忽视掉。正面情绪造成处理模式的变化，从而帮助创造性问题的解决，有利于解决由操作过程和界面设计产生出来的问题。换句话说，当感觉良好时，就会不太在意设计缺陷。愉悦的产品设计，其外表美观迷人，使用起来也就显得比较顺手，也比较容易。

产品使用起来得心应手会令人快乐，加倍喜欢。这是设计中为什么易用与情感总要联系起来的原因。正如唐纳德 · A. 诺曼（Donald A. Norman）所说的：美观的物品更好用。这要从辩证的方法来看问题：从正面来说，对于有压力的工作或者环境，优秀的易用性设计最为必要，要通过设计来尽可能让注意力分散和愤怒的情绪等最小化。在愉快、积极的情形中，人们比较可能容忍小困难和细枝末节的问题。换句话说，恶劣的设计虽不可迁就，但人们在愉快放松的时候，使人愉悦的设计会让人们更加能够容忍操作界面上的问题和困难。

然而从反面来说，一方面，操作方法复杂，难以识别的物品就算设计得再美观，造型多么优雅，也会影响用户对产品的喜爱。另一方面，丑陋的外表与冷漠的体态，也会因糟糕的情绪在如此简单、易用的产品操作中出现错误。这就是易用与情感的关系，相互依存，相互影响，共存于美好的产品中，能够解决人与物品之间的矛盾，改善人类的生活。

对于易用性和功能的认知研究固然非常重要，情感方面的研究也同样重要。易用与情感的并重将会使未来的产品能够：尽其功能、便于使用、富于乐趣。本书的观点是优良设计意味着平衡兼顾美学和易用。缺少了易用性，不管这个物品具有深层次的美感，还是仅表面上的漂亮都是一样的。最重要的是愉快地拥有并使用。

易用需要从认知、思维方面设计用户的操作行为，情感需要从价值观念的角度设计用户的审美、需求与愿望。易用表现的是物品对人的适应性、用户对物品的主动性，需要以思维、意向为目的，通过调查人的行为特性建立用户模型；情感表现的是个人追求、向往与憧憬的美好，需要以文化、社会观念为根基，通过调查人的心理需求，建立用户模型。易用与情感的共同之处皆在避免人与物品之间的鸿沟，建立人与物品之间的友好界面。总之，易用与情感的相互作用与结合，体现了“以人为本”的设计思想。

10.2 Web 易用性测试

与传统软件系统有所不同，Web 站点评估面临更大的挑战。大多数其他软件系统具有有限的任务和可预测的用户，而浏览 Web 站点的用户是异质的，各有各的原因。在日益增长的以 Web 为中心的消费环境中，用户到达 Web 站点的驱动因素有很多，也许是为了寻找信息，也许是为了满足消费需求，也许只是冲浪等。用户的定义正变得更加模糊，因为任何人都可以达到 Web 站点。角色和目标的不同，使每个用户对 Web 站点的需求也都不同，这使得影响 Web 易用性测试的因素增多。

除此之外，Web 站点在设计、测试方面和传统的软件系统也有明显的差别。例如，用户连接到 Internet 上的设备千差万别，从电话分机到局域网、数字专线等。在 Web 站点的设计及易用性评估中，必须考虑到这些差异。为了使用户能以他们喜欢的方式在网页

之间导航、浏览，Web站点在设计时就不能在用户的连接方式上加以过多限制。

因此，Web易用性测试，并不是将传统的软件系统的易用性测试方法加以简单的应用，而需要加入新的思想和方法。

10.2.1 Web易用性测试定义

Web易用性测试是使用科学的测试方法框架，对用户使用网站导航，在网站上完成若干任务等方面进行测试，测试者观察其行为并作记录，进行分析得出结论。网站易用性测试整个过程就是用户使用网站最初以及最真实的体验。所以通过易用性测试，可以了解到各个代表性的目标用户对网站界面、功能、流程的认可程度，获知改良的可能性方案，特别在交互流程中能得出一些很不错的用户行为规律。

人们对于Web易用性的含义并不都是非常明确的，可能就觉得“易用性”网站就是直观好用的、界面友好的，对此，Thomas A. Powell在《Web Design：The Complete Reference》书中对Web易用性给出了定义：

易用性是指在特定的使用环境下，一个站点可以被一组用户有效、高效且满意地达成某个目的所能达到的程度。

在这个定义中可以看出，在讨论网站的易用性的时候，首先要注意的是：谁是目标用户。因为对于不同类型，不同目的的网站来说，它的目标用户都是不同的。在限定了网站的目标用户之后，才能明确应该提供什么样的信息。而“有效性是指用户是否能够达成他们的目标，如果用户不能或仅是部分能够通过某个站点完成他们事先想要完成的事，这样的站点实际上是不可用的。”高效是指用户能够有效地完成预期的工作，而不会因为访问网站使工作效率降低。用户的满意则是网站建立的最终目的。

10.2.2 Web易用性测试的必要性

现在很多网站开发人员往往出于时间、经济上的压力，只追求在尽可能短的时间内用尽可能低的成本发布一个站点或对网站进行改进，忽略了网站开发前必要的需求收集和易用性测试，往往容易造成开发出的站点用户觉得不好用甚至放弃，后期改进难度大工作量大的尴尬局面。美国15个大型商业网站的调查显示，用户在寻找指定信息或在网站完成指定任务的成功率仅为42%，而约30%的用户因为浏览商品或购物流程的不友好而中途放弃购物车，这足以说明网站易用性对用户的行为影响。而在早期引入即使是最基本的易用性测试，都会给网站和用户带来很多收益。

在网络经济中，由于网站在开发和运行方式以及面向用户群体等方面的特点，网站易用性显得更加重要。首先，现有的网站开发技术使得建设一个网站在技术上变得比较容易，这就为那些缺少经验的开发者在短时间内草率地开发和建立网站提供了条件，降低了人们对质量期望的重视程度，导致出现了一大批易用性质量低劣的网站。而与此同时，网站的用户群体却在地域、文化背景、语言、受教育程度、计算机技能、年龄、性别、职业等方面呈现出空前的多样化，从而使网站在满足不同用户的要求上变得更加困难。在如此庞

大的网络世界中，用户比以前有了更多的选择。他们为什么要把时间浪费在那些混乱的、缓慢的或是不能满足他们需要的站点上呢？正是由于选择众多，而且另寻他处相当便利，所以网络用户明显表现出缺乏耐心，而且需要能迅速满足自己的要求。据统计，当用户首次访问网站留下不良印象时，40%的用户不会再次访问该网站。

另外在传统的实际产品的发展过程中，用户在购买产品之前，是无法体验其易用性的。然而网络扭转了这种局面。现在用户在承诺使用一个站点，并且准备花钱购买之前，就可以体验该站点的易用性。其中的区别是：在传统的产品设计和软件设计中，客户先付钱，后体验易用性。而在网络经济中，客户先体验企业网站的易用性，后付钱。同样对于其他类型的网站，要想实现网站的经济效益，就应让用户感受到良好的易用性质量。

由此可见，网站的易用性问题对网站发展会产生不容忽视的负面影响。对用户来说，会造成时间和金钱上的浪费，降低工作效率，增加挫败感；对网站拥有者来说，会导致失去用户，损失用户重复访问率，增加网站维护成本，增加培训和技术支持需求，降低企业声誉和竞争力。

10.2.3 Web易用性测试原则

易用性的判断受个人因素影响比较大，为了避免易用性测试的模糊性和主观性，这里需要提出测试原则并进行分析：

1. 每一块栏目的功能性应该显而易见

按钮、链接应该明显直观，避免花哨的装饰、过多的文字、错乱的排版，而影响阅读者的判断。

检查页面是否产生大量影响阅读信息的时候，需要针对网站功能列一个信息传达清单，以检查页面是否能正确达到传达效果。

在创新的、开拓性的或非常复杂的页面设计时，内容并不会一下被人理解，但页面应该从外观、元素、名称、布局、文字等方面，拥有自我解释的能力，让用户明白制作者的意图。

2. 用户浏览网页的原则

① 用户是以扫描的方式浏览网页，而不是细读（新闻、报告、产品描述等叙述性信息一类除外）。

原因是：

- 用户处在忙碌状态。
- 用户不必阅读所有文字。
- 阅读中的习惯以扫描的方式，在网页上看到什么取决于其想看到什么。

② 用户使用网站的时候，往往不是使用最佳的方式，而是选择符合他生活经验的合理方式，即满意即可。

原因是：

- 都很忙。

- 使用过程中有错误，也不会产生严重后果。
- 在设计不佳的网站，权衡选择并不会改善太多。
- 猜测比衡量省事。

③ 用户使用网站时，并不想弄清楚整个网站系统，而是处在勉强状态。

原因是：

- 网站的整个系统对用户来说并不重要，他需要的只是能正常使用。
- 如果发现某个事物能用，用户会一直使用，很少人去选择更好的方法。

3. 页面应该适合为扫描阅读

适合扫描阅读的页面应该具有以下5点：

(1) 建立清楚的视觉层次

视觉层次清晰的页面的特点：

① 越重要的部分越清晰。突出重点的方式有：字体特别，字号大，颜色特别，旁边有大部分留白，接近顶部等。

② 逻辑上相关的部分在视觉上也相关。例如把相近的内容分成一组，放在同一部分，采用类似的显示方式，或者把它们全部放在一个定义明确的区域。

③ 逻辑上包含的部分在视觉上进行嵌套。

(2) 符合用户阅读的习惯用法

习惯用法因为有用才会成为习惯用法。适当运用习惯用法，会使用户在网站间的访问更容易，保证了熟悉感。

习惯用法对设计师是种挑战。因为设计师面临的是创造性的工作，很多人想放弃习惯性用法。可这样有时候并不划算，不合适。习惯用法应该有选择性地使用。

一种新的习惯用法应该具备：

① 让用户清楚所创造的新方法。

② 拥有巨大的价值，值得用户去学习。

如果打算创新，那么必须理解准备换用的方法的价值，而很多设计师都低估了习惯用法提供的价值。

(3) 页面内容被划分成明确定义的区域

把页面划分成明确定义的区域很重要，因为这可以让用户很快决定关注页面的哪些区域，或放心地跳过哪些区域。

(4) 页面标识明显

明显标识可以用鼠标单击的地方。

(5) 降低视觉噪声

视觉噪声有两种：

① 眼花缭乱的页面。

② 背景噪声。

视觉噪声大都是与主要内容用相同或相似的颜色和装饰，使得扫描阅读时，难以阅读。

4. 让网页上的选择无须思考

三次无须思考，明确无误的单击相当于一次需要思考的单击。如果需要一直在网络上进行选择，那么让这些选择变得无须思考是让一个网站容易使用的主要因素。

5. 文字尽量简练

省略不必要的文字，网页上的文字尽量简洁，不使用欢迎词和指示说明。

10.2.4 Web 易用性测试标准

国外已有从用户的角度出发，应用易用性方法，对电子商务网站、政府网站和学术站点等不同类型站点进行比较深入的易用性分析和研究，产生了较为成熟的研究成果和可行的 Web 易用性测试指南和标准。这里介绍如下：

1. 网站易用性

美国的计算科学工程西北联盟（Northwest Alliance for Computational Science & Engineering，NACSE）从网站设计、网页设计和导航帮助 3 个方面制订了通用的网站易用性指南，这是改善和提高网站易用性的一组设计准则。针对网站易用性问题，从不同的角度提出了大量的易用性设计准则，如设计过程及考虑、内容及内容组织、标题或头文字、页面长度、页面布局、字体和文本大小、阅读和浏览、链接、图形、搜索、导航、软件和硬件、可访问性等。遵循这些通用的指南，可以有效改进网站易用性。

2. Nielsen 法则

Nielsen 法则是由易用性研究大师 Nielsen 提出来的，他从理论、方法、实践等各方面对网站易用性进行了研究，他认为网站易用性是由遵循一系列协议所获得的系统状态，并且网站的易用性缺陷能很容易被发现，他认为执行仅 5 项用户测试就可以发现 85%的问题。Nielsen 还认为网站具有同质性，如果用户花更多的时间浏览其他网站，那么也应该按照其他站点的模式设计自己的站点。按照这一观点，Nielsen 和同事测试了 20 个 B2C 电子商务网站，要求 64 个来自美国和丹麦的参与者在这些网站进行购物活动，基于研究过程中的观测记录以及专家的经验，Nielsen 等获得了创建更好的电子商务用户体验的 207 条易用性设计指导规则（后来被称为 Nielsen 法则）及一系列 B2C 电子商务网站易用性的评估报告。这些指导规则涵盖了大范围的主题，包括销售策略、信任感、分类页面、搜索、产品页面、检验与注册以及国际性用户。Nielsen 法则对如何提高和评价电子商务网站易用性具有指导性意义。

3. 微软易用性指南

本文采用的易用性评价指标体系是国际 IT 市场的领头羊——微软公司的微软易用性指南（Microsoft Usability Guideline，MUG）。MUG 围绕 5 个主指标形成了对 Web 站

点进行启发式评价的基础，这 5 个主指标是：内容、易使用性、促销、定制服务、情感因素。可以认为，这些指标覆盖了与 Web 站点的易用性有关的所有特征。

① 内容(Content)：用来评估 Web 站点所包含的信息以及将这些信息传递给用户的价值、能力。

② 易使用性(Easy of Use)：指的是使用 Web 站点时对用户的能力上的要求。站点对用户能力的要求越低，该站点则越易使用。如果一个网站必须是经过某些专门培训的人才能使用，那么该网站的易使用性是低的。

③ 促销(Promotion)：指 Web 站点在 Internet 或其他媒体上的广告宣传能力。一个网站的促销能力对促进网站的交易是很关键的。

④ 定制服务(Made for the Medium)：指 Web 站点能满足特定用户需求的能力。网络提供了为客户量身定制服务的空间，个性化是 Web 站点的关键需求。事实上，当今的市场营销策略，如关系营销、个人对个人营销等，都要求 Web 站点不能设计成静态的，而应当能提供动态的、能满足特定用户独特需求的内容。

⑤ 情感因素(Emotion)：指 Web 站点能作出情感反应的能力。实践表明，软件系统能像人一样有情感地做出反应，在计算机使用环境中起着非常重要的作用。

MUG 提供了一套进行 Web 站点易用性评估的指标，但并不是所有的指标都同等重要。对 Web 站点的易用性评估与 Web 站点所处的行业、用户浏览 Web 站点时的角色、用户的目的等因素密切相关。例如对儿童站点的易用性评价中，儿童“易用性”指标就应占有重要成分。因此，在对具体的网站进行评估之前，首先要确定这些评估指标的相对重要性。在实践中，确定重要性和次要性指标的方法是通过分配不同的分值来表示。

4. CIF 标准

CIF(Common Industry Format)是由美国国家标准技术局(NIST)组织制定的 IUSR (The Industry Usability Report)计划中首次提出。2001 年被正式颁布成为 ANSI/NCITS 345 号标准，作为易用性测试的标准报告。CIF 报告所提供的是采购商在评价产品易用性质量时所需的最基本的信息，在对具体产品使用 CIF 报告的时候可以增加内容。CIF 报告的基本内容包括产品说明、测试目的、测试对象、测试任务、测试环境、测试的方法和过程、指标体系和数据采集及数据分析结果等，其中指标体系包括了有效性、效率、满意度等。CIF 是对测试方法和测试结果进行详细说明的标准格式。CIF 报告对易用性测试起到了指导和规范的作用，实际上给出了一种易用性测试的标准，适用于软件产品和网站。

5. UMM 模型

国外又提出了 UMM(Usability Maturity Model，易用性成熟度模型)，它是 ISO15504 标准的一部分。它的宗旨就是从软件开发的组织体制上真正实现以用户为本。UMM 分为 6 个等级：从最初的等级 0“未知阶段”开始，经过等级 1“认知和个体执行”、等级 2“评价和管理”、等级 3“建立和实现”、等级 4“整合和预测”，最后到最高等级 5“制度化和优化自适应”的过程。它最终建立起一个可预测的过程，可以调整它的性能去满足现在和以后一段时间的客户需求，可靠地满足它所描述的目标。

此外还有IBM公司的用户接口构建指南(User Interface Architecture,UIA)以及苹果公司的人机接口指南(Apple Human Interface Guidelines,AHIG)等。当然这些标准或准则只是一些原则性的建议,在具体开发中也不能死搬硬套,而要根据具体情况灵活选用。

10.2.5 Web易用性测试支持工具

Web易用性支持工具,可以自动地检查、获取、分析、评估网站潜在的易用性问题,有的工具还可以提出相应的解决方案,甚至自动修改易用性问题,从而帮助开发人员迅速提高网站易用性。网站易用性支持工具大致可以分成3类:

① 评估网站实现技术的工具。这类工具主要是基于一组易用性评估指南,评估组成网站的各个页面的HTML代码,以确定它是否遵循HTML标准;检查如背景色、图像、字体、导航元素的一致性等。这类工具可以识别诸如可访问性、表单使用、性能、可维护性、导航、可读性等方面的易用性问题,产生页面易用性问题详细报告、潜在易用性问题优先顺序列表,有的工具还提供相应的编码修改建议,并附有相关的易用性原理,甚至可以直接修改评估报告中的HTML问题,使网站易用性得到迅速提高。这类工具主要有美国国家标准及技术研究所开发的WebSAT网站静态分析工具、IBM公司开发的AppScan网站易用性支持工具等。

② 基于智能浏览代理的工具。这类工具使用一个统计模型模拟用户经验,利用智能浏览代理(Intelligent Browsing Agent)遍历网站以收集关键的统计信息,从可访问性、装载时间、内容三个方面对网站进行评估,可以用一个快速的数字化分析评定网站的等级,提供性能基准,但不提供设计修改建议。这里可访问性是通过确定浏览路径,即从开始页到任何给定页的最短点击数,计算访问时间来描述;装载时间是计算站点页面装载的平均时间;内容则是指汇总组成站点的不同媒体元素(文本、图像、声音等)和客户站点技术(如Flash、DHTML等)的百分比,可以与竞争者做相应的比较分析。这类工具中最著名的是WebCriteria公司(www.webcriteria.com)的MAX软件。

③ 用户反馈收集和分析工具。这类工具一般由在线工具组成,拥有针对网站使用质量而精心设计的用户调查问卷,帮助收集用户反馈意见和建议,从而识别网站易用性问题,也可以进行市场调研。这类工具是最好的、最容易使用和最有效的易用性支持工具,因为它们是基于最终用户直接反馈以识别易用性问题,也真正体现了"以用户为中心"的原则。典型工具的有NetRaker公司开发的NetRaker套件、WAMMI问卷等。

10.3 易用性测试实践

10.3.1 易用性测试方法

易用性测试的具体方法非常丰富,Santon和Young在文献回顾基础上总结出了60余种易用性测试方法,图10-1列出了一部分方法。易用性测试方法可归结为主观和

客观两大类。主观测试主要采取问卷、讨论和会议的方式搜集用户的诉求，适用于产品设计的不同阶段。客观测试主要是采取在实验室或现场分析仪器记录数据的方式，适用于原型和成型产品的评价。

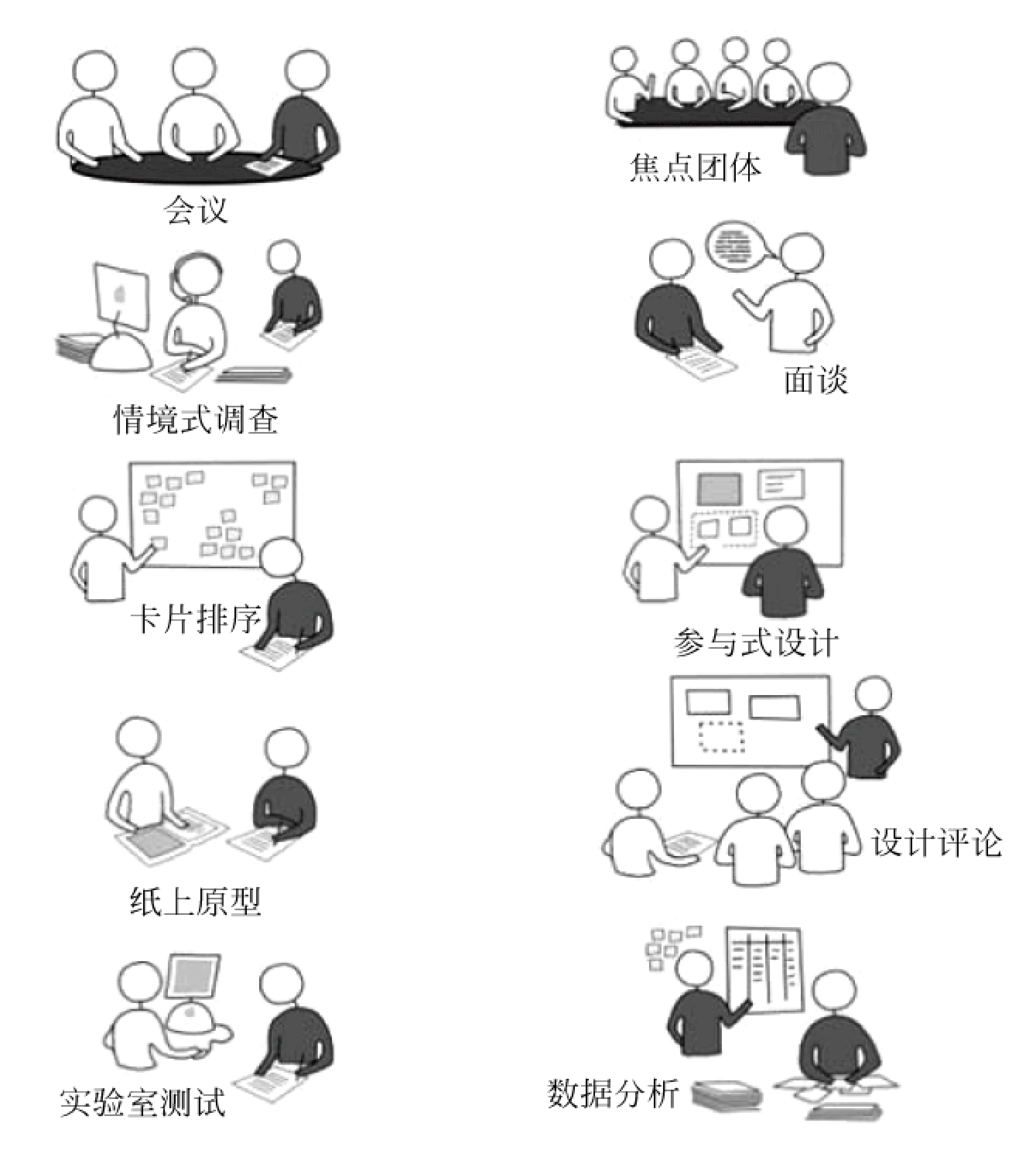


图 10-1　易用性测试方法

1. 主观测试方法

(1) 用户调查法

用户调查法是直接询问用户的一种方法，是社会科学研究、市场研究和人机交互学中沿用已久的技术，适用于快速评估、易用性评测和实地研究，以了解事实、行为、信仰和看法。用于易用性评测研究的是用户如何使用系统以及哪些功能是用户非常喜欢或不喜欢的。这种方法尤其适用于客观上较难评测的、与用户满意度相关的问题。用户调查法有如下两种：

① 访谈法。访谈与普通对话的相似程度取决于待了解的问题和访谈的类型。在访谈期间，采访者可以自始至终地分析受访者对各个问题的回答，一旦发现问题被误解了，可以立即用不同的方式提出。访谈的方式可以是面对面的交流，也可以通过电话进行，网络聊天的形式也是有效的。进行电子商务的易用性评测时，以上 3 种方式均可以被采用。访谈有 4 种主要类型，即开放性(或非结构化)访谈、结构化访谈、半结构化访谈和集体访谈。具体采用何种访谈技术取决于评估目标、待解决的问题和选用的评估范型。例如，如果目标是大致了解用户对信息的理解程度，那么非正式的开放式访谈通常是最好的选择；

但如果目标是搜集关于特定特征(如能够辨认出多少功能)的反馈,那么结构化的访谈调查更为适合,因为其目标和问题更为具体。访谈主要用于收集一些指标的计算参数,如上面提到的用户能够辨认出的功能数、对信息的理解程度等。半结构化访谈和集体访谈没有在评测电子商务的易用性中被使用。

② 调查问卷。它是用于收集统计数据和用户意见的常用方法,可以是纸质印刷品,也可以是计算机环境下的交互调查问卷。它与访谈有些相似,也是用来了解用户的满意度和遇到的问题,但是最大的不同在于其用户可以在不需要评测人员在场的情况下独立填写,只要确保措辞明确,避免可能的误导,问卷可以是开放式的问题,也可以是封闭的问题。但某项研究发现,用户回答提供选项问题的准确率为85%(与观察到的用户实际情况相比较),而用户回答没有列出可选择的描述项目的开放式问卷的准确率只有48%。因此,为了保证所收集的数据有较高的可信度,电子商务的易用性评测使用封闭问题,给出等级尺度来表示用户对系统一些方面的喜欢程度。使用调查问卷方式评测的指标有界面的易懂程度、操作的顺畅性、操作的便捷性、页面的吸引程度。

普渡大学可用性调查表

使用说明:本调查表共有100题,回答每一个问题时请按照如下3个步骤:

① 请评估每一个问题是否适用于所评审的系统,如果不适用,跳到下一题,如果适用,请继续回答下面两个问题。

② 对于所评估的系统,请评价该问题的重要性(1是最不重要的,3是最重要的)。

③ 评价系统的表现(1是非常糟糕,7是非常好),如果不存在,请选择不存在该项。

一、兼容性

1. 光标的控制是否符合光标的移动?
2. 用户控制的结果是否符合用户的期望?
3. 所提供的控制是否符合用户的技能水平?
4. 界面的编码(如颜色、形状等)是否为用户所熟悉?
5. 用词是否为用户所熟悉?

二、一致性

6. 界面颜色的编码是否符合常规?
7. 编码是否在不同的显示及菜单上都保持一致?
8. 光标的位置是否一致?
9. 显示的格式是否一致?
10. 反馈信息是否一致?
11. 数据字段的格式是否一致?
12. 标号的格式是否一致?
13. 标号的位置是否一致?
14. 标号本身是否一致?
15. 显示的方向是否一致?(漫游或卷动)
16. 系统要求的用户动作是否一致?
17. 在不同的显示中用词是否一致?

18. 数据显示和数据输入的要求是否一致？
19. 数据显示是否符合用户的常规？
20. 图形数据的符号是否符合标准？
21. 菜单的用词和命令语言是否一致？
22. 用词是否符合用户指导的原则？

三、灵活性

23. 是否可以使用命令语言而绕过菜单的选择？
24. 系统是否有直接操作的功能？
25. 数据输入的设计是否灵活？
26. 用户是否可以灵活地控制显示？
27. 系统是否提供了灵活的流程控制？
28. 系统是否提供了灵活的用户指导？
29. 菜单选项是否前后相关？
30. 用户是否可以根据他们的需要来命名显示和界面单元？
31. 系统是否为不同的用户提供了好的训练？
32. 系统是否可以自己改变视窗？
33. 用户是否可以自己命名系统命令？
34. 系统是否允许用户选择需要显示的数据？
35. 系统是否可以提供用户指定的视窗？
36. 为了扩展显示功能，系统是否提供了放大的功能？

四、可学习性

37. 用词是否清晰？
38. 数据是否有合理的分类，易于学习？
39. 命令语言是否有层次？
40. 菜单的分组是否合理？
41. 菜单的顺序是否合理？
42. 命令的名字是否有意义？
43. 系统是否提供了无惩罚的学习？

五、极少化的用户动作

44. 系统是否为相关的数据提供了组合输入的功能？
45. 必要的数据是否只需要输入一次？
46. 系统是否提供了默认值？
47. 视窗知觉的切换是否容易？
48. 系统是否为经常使用的控制提供了功能键？
49. 系统是否有全局搜索和替代的功能？
50. 菜单的选择是否可以使用点击的功能？（主要的流程控制方法）
51. 菜单的选择是否可以使用键入的功能？（辅助的控制方法）
52. 系统是否要求极少的光标定位？

53. 在选择菜单时，系统是否要求极少的步骤？
54. 系统是否要求极少的用户控制动作？
55. 为了退到更高一级菜单中，系统是否只需要一个简单的键入动作？
56. 为了退到一般的菜单中，系统是否只需要一个简单的键入动作？

六、极小化的记忆负担

57. 系统是否使用了缩写？
58. 系统是否为输入分层次的数据提供了帮助？
59. 指导信息是否总是可以得到？
60. 系统是否为序列的选择提供了分层次的菜单？
61. 被选的数据是否有突出显示？
62. 系统是否为命令提供了索引？
63. 系统是否为数据提供了索引？
64. 系统是否提示在菜单结构中的当前位置？
65. 数据是否保持简短？
66. 为选择菜单使用的字母代码是否经过认真的设计？
67. 是否将长的数据分成不同的部分？
68. 先前的答案是否可以简便地再利用？
69. 字母大小写是否等同？
70. 系统是否使用短的代码而不使用长的代码？
71. 图符是否有辅助性的字符标号？

七、知觉的有限性

72. 系统是否为不同的数据类别提供不同的编号？
73. 缩写是否清晰而互补相同？
74. 光标是否不同？
75. 界面单元是否清晰？
76. 用户指导的格式是否清晰？
77. 命令是否有清晰的定义？
78. 命令的拼写是否清晰？
79. 系统是否使用了易于分辨的颜色？
80. 目前活动的窗口是否有清楚的标识？
81. 为了直接比较，数据是否成对地摆在一起？
82. 是否限制语音信息使用的数量？
83. 系统是否提供了一系列相关信息？
84. 菜单是否和其他显示信息有明显的区别？
85. 颜色的编码是否多余？
86. 系统是否提供了视觉上清晰可辨的数据字段？
87. 不同组的信息是否明显分开？
88. 屏幕的密度是否合理？

八、用户指导

89. 系统反馈的错误信息是否有用？
90. 系统是否提供了“取消”的功能？
91. 错误的输入是否被显示出来？
92. 系统是否提供了明确的改正错误的方法？
93. 系统是否为控件输入提供了反馈？
94. 是否提供了“帮助”？
95. 一个过程的结束是否标志清楚？
96. 是否对重复的错误有提示？
97. 错误信息是否具有建设性并提供有用的信息？
98. 系统是否提供了“重新开始”的功能？
99. 系统是否提供了“撤销”的功能？
100. 用户是否启动流程控制？

可用性分数：$\sum(w_i * (S_i - P_i))/7 * \sum(w_i * I_i) * 100$

其中：

i：第 i 个问题。

S_i：该系统在第 i 个问题上所得的分数。

P_i：=1，如果第 i 个问题适用但不存在。

P_i：=0，如果第 i 个问题不适用。

I_i：=1，如果第 i 个问题适用。

I_i：=0，如果第 i 个问题不适用。

w_i：第 i 个问题重要性的得分。

（2）专家评审法

专家评审分为以下两种：

① 启发式评估。它是由 Jakob Nielsen 和他的同事们开发的非正式易用性检查技术，使用一套相对简单、通用、有启发性的易用性原则来进行易用性评估。其具体方法是，专家使用一组称为启发式原则的易用性规则作为指导，评定用户界面元素（如对话框、菜单、工具条、在线帮助等）是否符合这些原则。在进行启发式评估时，专家采取角色扮演的方法，模拟典型用户使用产品的情形，从中找出潜在的问题。由于启发式评估既不需要用户参与，也不需要特殊设备，所以其成本相对较低，而且较为快捷，因此该方法也被称为经济评估法。

② 走查法。它是从用户学习使用系统的角度来评估系统的易用性。这种方法主要用来发现新用户使用系统时可能遇到的问题，尤其适用于没有任何用户培训的系统。走查就是逐步检查使用系统执行的过程，从中找出易用性问题。走查的重点非常明确，如用于评估系统的功能数目，可以获得帮助的程度等。

2. 客观评测方法

(1) 用户测试法

易用性既然是评价软件质量的标准，而且是从用户角度出发，评价起来当然少不了用户的参与，在所有的易用性评估法中，最有效的应当是用户测试法。该方法是在测试中，让真正的用户访问某网站，如 CNN 网站，以评估该网站的易用性，而测试人员在旁边观察、记录、测量，并有可能还用到眼动仪、动作分析仪等实时监控设备，如图 10-2 所示，这是眼动仪记录下的眼动轨迹。因此用户测试法最能反映用户的需求。

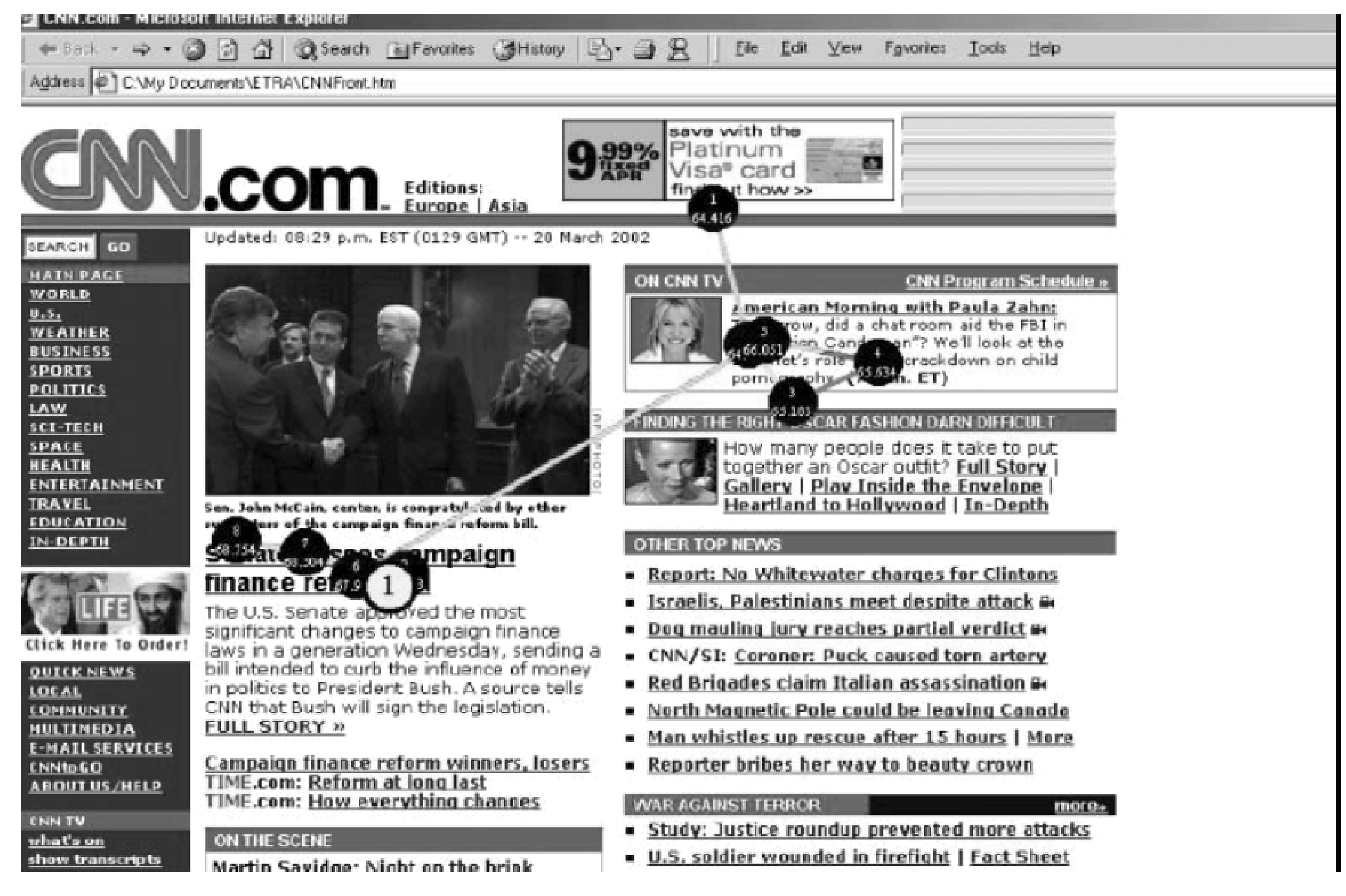


图 10-2　网页评估眼动轨迹

根据测试地点的不同，用户测试可分为以下两种：

① 实验室测试。它是在易用性实验室中进行的，也是在一个受控环境下执行测试。用户被引入系统并且要求根据预先设置的场景执行几个关键的任务。易用性测试可能在一个真实的系统上执行，或是一个演示(例如 PowerPoint)，它们只展示被测试系统中的元素。用户的活动使用两个摄像机录制下来：一个录制屏幕上的活动，一个录制用户的反应和表情。另外易用性专家监视易用性测试，记录感兴趣的项目。如图 10-3 所示的软件操作时，一个摄像机录制软件操作活动，一个摄像机录制操作者的表情、语言等。

② 现场测试。它是由易用性测试人员到用户的实际使用现场进行观察和测试。和实验室测试相似，只不过在现场执行。它通常在系统或环境很复杂，以至于很难在实验室中复制的情况下操作。现场测试可能也被用于研究在真实环境下的用户表现。现场测试使研究工作在他们典型工作环境系统的用户变为可能。这种类型测试的好处是让用户忽略在进行正式测试的感觉。

根据试验设计的方法不同，用户测试可以分为有控制条件的统计实验和非正式的易

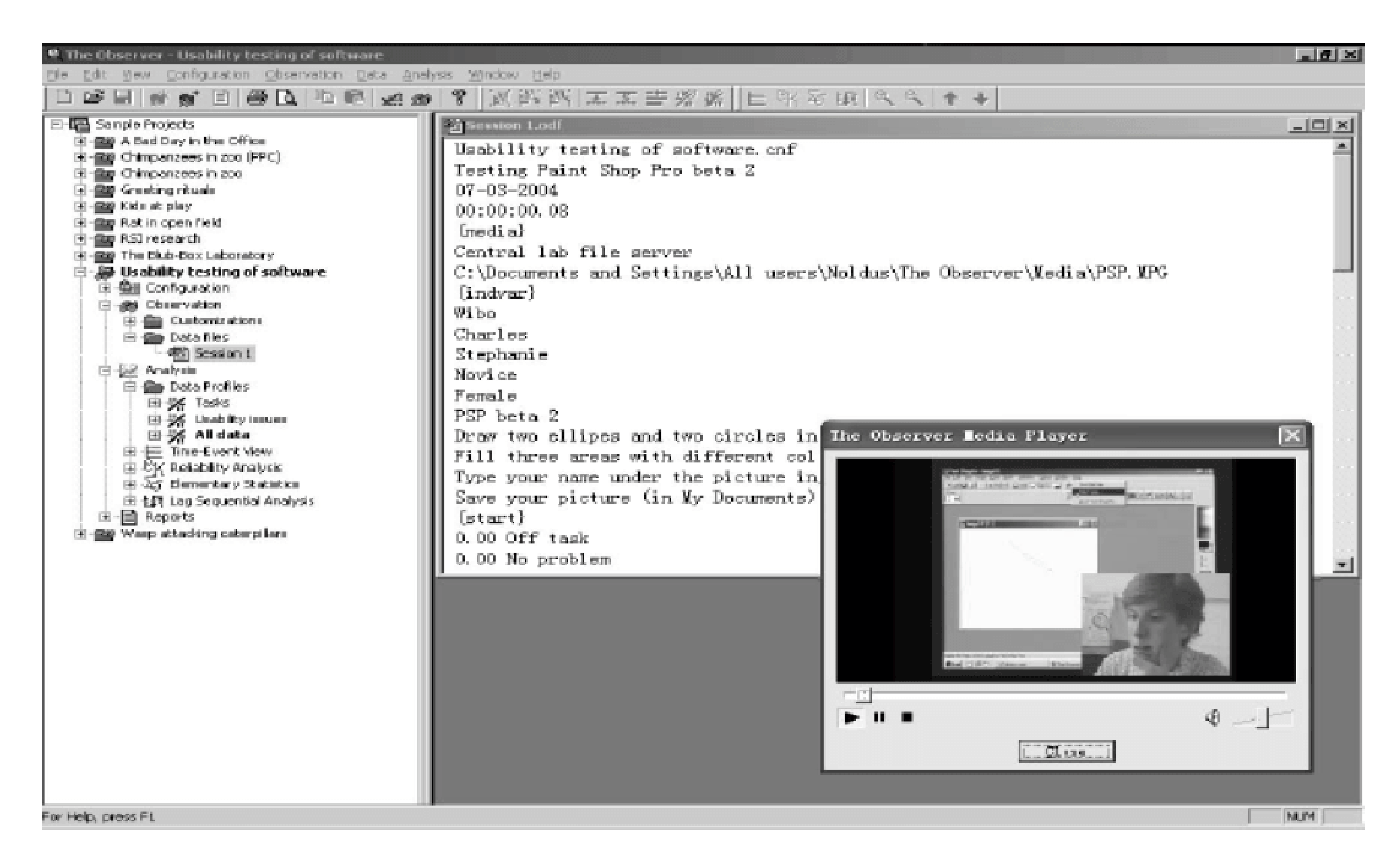


图 10-3　软件的实验室测试

用性观察测试。这两种实验方法在某些情况下也可以混合使用，所以经常被笼统地称为易用性实验。易用性的实验就是在产品实际应用的环境之外，就特定的环境、条件、使用者进行测试，以记录系统的表现，能对特定的因果关系进行验证，得到量化的数据。在软件易用性评测中，对于用户操作时间、寻求帮助的频率、出错情况等数据的获得，就是使用的这种用户测试法。因为不同用户的时间和空间的限制，实验室测试和现场测试均被使用。

用户测试还有其他常用的方法，有焦点团体讨论(Focus Group Discussion)及发声思考(Thinking Aloud)等。

- 焦点团体讨论是一般市场营销研究常用的手段。邀请一群使用者，一般 5～8 人一起就几个焦点问题进行讨论，由一位主持人掌控讨论的方向，围绕着预定的题目进行，让参与者都能畅所欲言地热烈讨论。不过若针对软件进行讨论，必须要考虑系统的规模与使用的体验，对企业的软件来说，一次讨论绝对不够，必须要进行一系列的讨论与评价。
- 发声思考法是心理学研究所用的研究方法，在国外被人机交互或易用性的研究者用来评估软件的使用。发声思考法要求受测者使用指定的系统，边用边说话，说出使用之时心中想的一切，包括困难、问题、感觉等。这个方法能从每位受测者的评价过程中收集到相当大的信息，而所需邀请的受测者也不多，在国外的相关业界说它是标准的软件使用质量评价方法。

(2) 自动化工具测试法

自动化测试是手工测试的增强，测试活动的自动化在许多情况下可以提供其最大价值，自动化测试工具减轻了测试工作量并缩短了测试时间。虽然在易用性评测中，人(用户和专家)的主观意见和客观表现是评测的重要手段，但是使用静态分析工具代替专家执行遍历测试也是非常好的选择，其优点就是全面覆盖、节省人力物力以及准确性高。在电

子商务易用性评测中，测试页面的易用性就可以使用自动化测试工具。测试页面的易用性主要是考察链接。链接是 Web 应用系统的一个主要特征，它是在页面之间切换和指导用户去一些不知道地址的页面的主要手段。链接测试可分为 3 个方面：

① 测试所有链接是否按指示的那样确实被链接到了该链接的页面上；

② 测试所链接的页面是否存在；

③ 保证 Web 应用系统上没有孤立的页面，所谓孤立页面是指没有链接指向该页面，只有知道正确的 URL 地址才能访问。

链接测试可以自动进行，现在已经有许多工具可以采用，如 Pure_load、Webcheck。

以上介绍的易用性测试方法都是多年工业实践证明切实有效的。提倡在易用性测试中采用一种结合主观和客观测试的综合方法。在各个方法的实际运用中，可以根据具体情况对方法执行上的某些细节灵活掌握。因为设计易用性测试一般要考虑到软件的设计情况，在特定的产品开发项目中，如何选择所使用的易用性测试方法，直接关系到易用性工程的运用效果。在这里一定要综合考虑开发过程当时所处的阶段、各种方法所能提供的信息以及它们所需要的技能、人员、时间、设备等方面的资源，在此基础上，选择一组适合具体情况、能够互补和相互衔接的方法，使得以用户为中心的设计理念得到尽可能的充分体现。

10.3.2 易用性质量指标体系

评估和改进产品的易用性质量，需要有一种客观、统一和定量的衡量标准，然而怎样建立这样一种标准，一直是个难题。经过易用性工程界多年的不懈努力，对易用性衡量标准的看法逐渐趋于一致，即易用性是特定产品在特定使用环境下为特定用户用于特定用途时所具有的有效性(Effectiveness)、效率(Efficiency)和用户主观满意度(Satisfaction)。这里的有效性、效率和满意度 3 个指标往往是通过用户评估或测试来获得的。这一定义已被纳入 ISO 9241—11 国际标准，美国的 CIF 易用性测试报告标准也采用了这一定义。

1. 有效性指标

有效性指用户完成特定任务和达到特定目标时所具有的正确和完整程度。一般是根据任务完成率、出错频度、求助频度这 3 个主要指标来衡量的。

(1) 完成率(Completion Rate)

根据任务性质的不同，完成率指标的含义可以有以下两种：

① 当任务不可分，即只有完成和未完成任务两种状态时，完成率为完成任务的用户所占的百分比。

② 如果任务可分，即存在部分完成任务的情况时，用户有效完成的工作占该任务的比例称为目标实现率(Goal Achievement)。例如，某任务是让用户使用绘图软件画出 3 个不同的几何图形，那么该任务的目标实现率就应取决于用户所画出图形的数量，如果画出了 4 个，则目标实现率应为 80%。如果考虑到各图形复杂程度的差异，还可以给各图形赋予不同的权重。因此在任务可分时，任务完成率应为用户的目标实现率。

(2) 出错频度(Errors)

出错频度是通过用户执行某个任务过程中发生错误的次数来衡量的。

(3) 求助频度(Assists)

这是指用户在完成任务过程中遇到问题而无法进行下去时,求助于他人或查阅联机帮助或用户手册的次数。在提供任务完成率指标时,应区分有帮助和无帮助情况下的完成率。

2. 效率指标

效率指的是产品的有效性(完成任务的正确完整程度)与完成任务所耗费资源的比率。这里的资源通常指时间,这时的效率为单位时间的工作量。在相同使用环境下,用户使用效率是评定同类产品或同一产品的不同版本孰优孰劣的依据之一。

效率的计算公式为:

效率=任务有效性/任务时间

这里的任务有效性一般是用户的任务完成率,任务时间为用户完成任务的时间。效率刻画了用户使用产品时单位时间内的成功率。一个高效的产品应当可以让用户在较短时间内以较高的成功率完成任务。同样,对效率也应区分有帮助和无帮助两种情况。

3. 满意度指标

满意度刻画了用户使用产品时的主观感受,它会在很大程度上影响用户使用产品的动机和绩效。满意度指标通常使用问卷调查手段来获得。目前有多种广泛使用的标准问卷,如 SUMI、WAMMI、ASQ、PSSUQ、SUS 等,它们所采用的指标体系各有不同,例如 SUMI 问卷调查的综合满意度指标为 0~70,平均值为 50。

用户测试案例

微软公司为了评估和比较其操作系统产品的易用性质量,于 1999 年对它所开发的 Windows 98、Windows NT 4.0、Windows 2000 Professional Beta 2 和 Windows 2000 Professional Beta 3 这 4 个操作系统产品进行了一次比较全面的用户测试。为了保证评估结果的客观和公正,它委托专业可用性咨询机构 AIR,按照美国 GIF 可用性测试报告标准进行这次测试。

这次测试对每个产品选择了 36 个测试用户,其中生手、初级熟练者和中级熟练者(根据微软对用户的五级分类标准)各为 12 人,分成相应的 3 个组。所选择的测试任务为 22 个在 Windows 操作系统上常用的任务,如启动程序、保存文件、发送电子邮件、安装软件等,对每个任务都规定了完成的时间限度和成败标准。采用的可用性指标体系如下。

有效性:任务完成率=完成任务数/总任务数。

效率:所有任务的平均完成时间。

满意度:对产品设计、易学、易用、用户界面、易浏览、措辞、产品改进、购买意愿和无培训易用性这 9 个指标的评分。

测试是在 AIR 的可用性实验室中进行的,在测试过程中,测试用户按照书面任务说

明的要求独立完成预定的各个任务，测试管理人员通过实验室的单向镜观察用户的操作，记录任务完成时间和成败情况等数据，并对整个过程进行录像。测试结束后，对用户进行满意度问卷调查。最后对所有测试结果数据进行分析和处理，分别得出 4 个产品的有效性、效率和用户满意度指标上的比较结果和综合可用性比较结果。

10.4 易用性测试应用

国外的一些著名软件公司早在 20 世纪 70 年代末 80 年代初就已经意识到软件易用性的重要性，并开始这方面的研究和实践。如 IBM 公司早在 1970 年就引入了易用性测试，微软公司在 1988 年也开始进行易用性测试。这些公司花费大量时间和金钱探索设计软件用户界面的最佳方式。他们用上了由人体工程学专家掌舵的专业易用性实验室。这些实验室装备了反光镜和视频摄像机记录用户使用软件的方式。用户（主体）所做的任何行为，从按下哪个键，如何使用鼠标，到犯什么样的错误，对什么感到困惑，都加以分析，以提高用户界面设计。大家所熟知的微软 Windows 95 在推向市场前就经过了大量的易用性测试，从而保证了该产品除了具有强大的功能和稳定的性能外，还具有很强的易用性，能够为多数人所接受，这也是 Windows 产品为什么能在短时间内风靡全球的原因之一。

下面是一个比较典型的易用性测试的例子。

在 Windows 95 的关于窗口管理的设计过程中，开始在设计的最初阶段是将应用程序最小化的窗口放在 Desktop 上一个打开的 Plate 中（见图 10-4）。但是通过易用性测试发现，用户还是无法快速地返回先前最小化的程序（需要最小化当前打开的所有应用程序）。最后设计人员增加了 Taskbar（见图 10-5），并通过易用性测试证明了后面的方案是最佳的，也就是现在看到的这个样子。



图 10-4　Plate



图 10-5　Taskbar

那么进行软件易用性测试会不会增加软件开发的成本，延长软件的生产周期呢？IBM 和微软等公司认为，对开发商和制造商来说，一种具有易用性的产品的回报远远大于在它上面的投入。HP 公司在一种用于检测网络错误的软件应用易用性测试后，不仅没有延长产品的生产周期，而且在产品推出的 18 个月内就收回了投资。“1 美元的投入获得 10～100 美元的回报”。这是 IBM 公司对其创造的经济利润的描述。因此易用性研究在一些国际性大企业中越来越受重视，这是因为它能给企业带来巨大的、至少是稳定的经济回报。

易用性之所以能获得这么好的回报，有以下几个原因：

首先，易用性在一定程度上能降低产品成本。虽然看似简单的产品不一定易用，但这要比增加一些对用户来说无意义的功能强得多，而且不可否认“简单”是易用性产品的重

要特征之一。视觉上的简洁、功能上的明确，能提高人们接受产品的容易度。而这样的产品往往在制作成本上比复杂的产品低得多。其次，与通过降低价格提高竞争性的方式相比，提高产品的易用性这个途径对企业来说更具保障性。因为前者对企业自身来说肯定降低了回报，而后者不用降低价格，仅通过良好的用户经验就能吸引一批稳定的消费者。很多时候用户愿意花更多的钱以获得产品使用时的便利。另外，提高产品的易用性能使企业与用户靠得更近。与技术研究不同的是易用性研究侧重于对人的研究，只有对用户有充分地了解，才有可能获得准确的资料，设计出体贴用户的产品，而这样的产品往往会受到用户的欢迎和信赖。以网站设计为例，通过对用户使用过程的长期观察、发现，与设计师的初衷相反，用户不是阅读网页而是浏览，根据这个行为特征，设计师采取一些措施，如创建清晰直观的页面层次结构，利用惯例把页面分割成明确的不同区域，使视觉干扰最小化等，从而增加用户对网站内容的吸收量，提高网站的使用效率。

一个软件开发过程中，最初的工作是最为关键的，而软件设计正是这样一个关键的工作。软件易用性测试就是从产品的设计阶段就开始进行，采用以用户为中心的设计方法(User-Centered Design，UCD)进行系统设计，并开发出产品的用户界面的原型。这个原型可以是草图，也可以是一个原型化软件，然后用户根据这个原型进行易用性测试，设计员根据用户测试结果对设计进行修改，如此反复直到产品推向市场。它的流程可以用图 10-6 来表示。

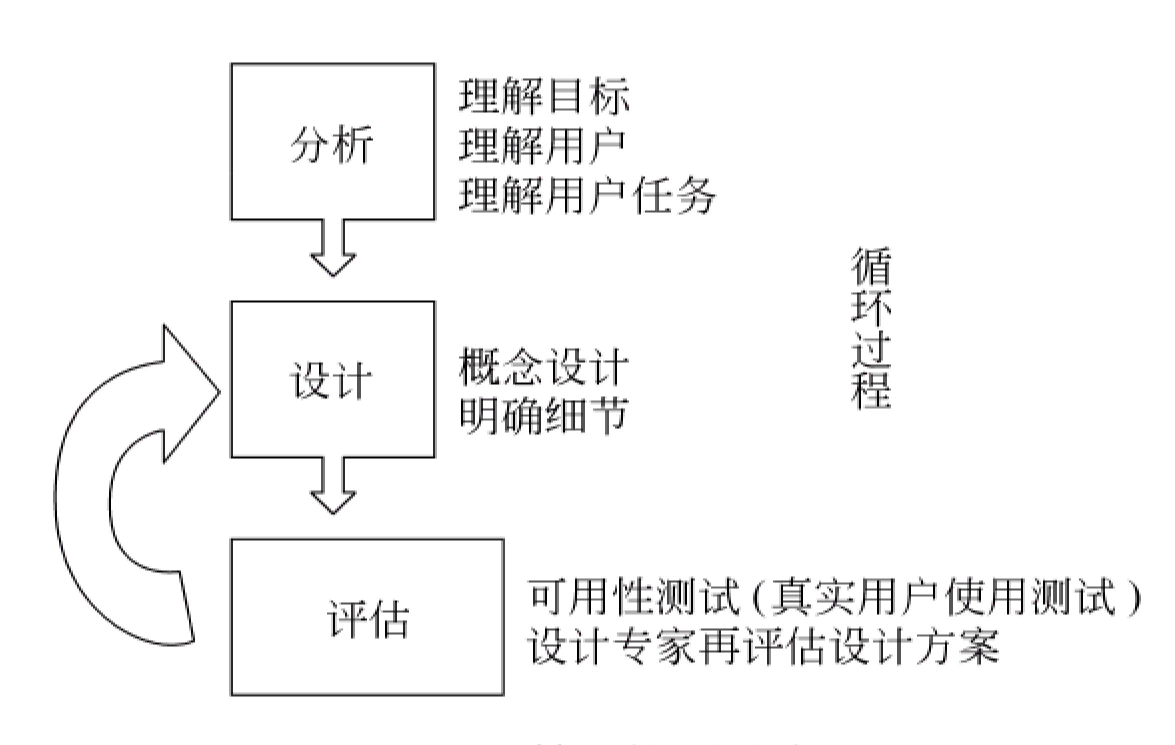


图 10-6　易用性测试流程

从上面这个易用性测试过程可以看到，易用性测试是在真实产品(服务)设计之前，和用户交互的必需的一步，以确保产品(服务)设计改进符合用户需求，说明测试是一个重复性过程，用户参与应该从早期阶段，而不是等产品(服务)就绪之后，这与 RUP 的最佳实践经验相吻合。一个软件从设计开始到测试结束一直有用户在参与，充分考虑了用户的需求。正是用户最大程度的参与，才使软件从根本上贴近了用户，符合用户的利益，从而保证了软件的高易用性。而由于软件的高易用性，用户在使用的过程中会明显地减少出错几率，因此可以降低开发商的支持和服务费用。所以进行软件易用性测试，不仅不会增加开发成本，在某种程度上还可以降低软件的成本。从另一个角度讲，易用性测试是在产品开发的早期就开始介入，这样就能够保证产品的易用性问题，在软件设计的早期就被发现并得到控制，避免了在后期进行修改而带来的各种资源的浪费，因此进行易用性测试也不会延长产品的生产周期。

可以套用时下流行的“双赢”说法来形容软件的易用性。对用户而言，它可以减少出错的几率，提高生产效率；从开发商的角度看，它可以增加用户满意程度、减少培训和服务费用。在日新月异的信息社会中，计算机正发挥着越来越重要的作用，软件的易用性问题也会变得越来越突出，它不仅是用户迫切关心的问题，也是每一个软件开发商必须面对和解决的问题。

实践证明，一个功能不完善或性能极差的软件产品是绝对不具有易用性的。但即使功能完善、性能指标较高的软件产品也不意味着具有很好的易用性。软件易用性与软件的功能、性能等指标，是相对独立又紧密联系的。因此，可以说软件产品的易用性与软件产品的功能、性能及可靠性一样，是影响软件产品生存周期的一个重要因素。

10.5 小　　结

在软件产品开发过程中，软件易用性的测试是必不可少的一环。易用性是从人的角度来看软件系统是否易用、高效，使人满意。在软件质量指标体系中，易用性有其重要的固有特性，是衡量软件质量的重要指标，是软件工程中的一个专门的研究领域。

易用性测试不仅是用于核对项目进度的一个里程碑，当最后一个参与者完成任务的时候，易用性测试还没有结束。整个团队必须仔细研究结果，设定优先次序，基于结果或网站原型进行修改。

现在，网站的数量和应用范围都在迅速发展，网站建设技术也日新月异，网站易用性问题更显得越来越重要。网站易用性方法论为提高网站易用性指明了方向，还需要加大对这种方法的推广应用。网站易用性设计指南还只是一些原则性的建议，有的还缺乏实际的可操作性；网站易用性支持工具目前主要侧重网站制作后期的检测和改进，还有待于进一步开发支持易用性工程各种方法的支持工具，另外现有的支持工具还存在着一定的局限性，并不能完全识别或解决网站潜在的易用性问题。

习题与思考

1. 简述易用性。
2. 优秀用户界面常见的 7 个要素是什么？
3. 简述易用性的原理。
4. 易用性测试的原则是什么？
5. Web 易用性测试的原则是什么？
6. 简单介绍 Web 易用性的测试标准。
7. 简单介绍 Web 易用性测试工具的种类。
8. 易用性测试方法的主观测试方法包括哪些？
9. 易用性测试方法的客观测试方法包括哪些？
10. 易用性质量指标包括哪些？

第11章 无障碍测试

无障碍测试就是无障碍理念的必然扩展，伴随着信息无障碍，尤其是无障碍 IT 产品设计、无障碍软件开发的发展而诞生的。

20 世纪 60 年代初，美国残疾人受民权运动的影响，他们联合起来为争取其基本权利而斗争，抗议社会对他们的歧视态度和不平等待遇，以及环境中的种种障碍给残疾人造成通行上的困难。在国际社会团体、社会各阶层的声援和倡导下，"无障碍"的概念开始形成，但那时的概念仅限于建筑及其环境的无障碍。

1974 年美国威斯康星州立大学教授 Ron Mace 在国际残障者生活环境专家会议中，第一次提出了通用设计的概念，他认为进行产品设计时，不单是生理层面的无障碍，还应包含心理层面无障碍的全人关怀设计。这是 IT 产品无障碍设计的前身。受此影响，IBM 公司于 1959 年成立了易用性实验室，接着成立了无障碍中心。但无障碍测试的概念到 2003 年由 IBM 公司美国无障碍中心研究员 Jim Thatcher 博士在他的书《Constructing Accessible Web Sites》中才首次提出。

无障碍测试脱胎于易用性测试，很多理论来源于易用性测试。对于无障碍的研究现在已经发展为一个独立的领域，无障碍功能已经成为软件和网络产品中不可缺少的一部分。目前，对于无障碍测试的研究，在国外已经纷纷开展起来了，在我国则刚刚起步。

无障碍的初衷是为了方便残疾人自由参与社会活动，随着美国学者 Ron Mace 将无障碍设计发展为通用设计，国际上无障碍理念已悄然发生改变。那就是认为无障碍不仅是残疾人的事，也是正常人在内的事。即无障碍环境建设不仅是局限于为残障人士提供帮助服务，而是将服务范围扩展到了所有的人群，只为残障人士提供服务的无障碍设施不是完整意义上的无障碍设施。根据有关定义，社会上大部分人都有各种程度的障碍，即使他们不愿视自己为残障人，或不愿如残障人一样生活。

正常人身体的某一部分机能，例如感知能力、行动能力、表达能力、思维判断能力都会随着年龄、所处的环境的改变而发生变化或衰退。今天的健康、正常人或许明天乃至以后就需要使用无障碍设施，这就是通用设计的初衷。

目前，国际上发达国家的无障碍建设都是以最广大的人群为服务对象。如美国的全能住宅研究、日本的环境标识系统研究等在满足残障人使用的同时，都考虑了正常人群在特定环境中的便利与需求。与此同时，随着时代的发展，网络和信息技术的普及，又产生了一个新名词：信息无障碍。可以相信，今后无障碍测试还会赋予新的内容，并将得到进

一步发展，呈现出更加丰富多彩的局面。

11.1 无障碍测试基础

11.1.1 无障碍测试的提出

21 世纪是信息和科技的时代，让残疾人能像正常人一样获取信息，是残疾人独立生活、公平参与及成功融入社会的最重要因素。2000 年冲绳八国首脑会议发表了旨在促进信息通信技术发展，缩小国家间、地区间信息技术发展差距以及建设全球信息化社会的共同宣言——《关于全球信息社会的冲绳宪章》，即 IT 宪章，它第一次提出了信息无障碍这个概念。

IT 宪章指出，信息通信技术是 21 世纪社会发展的最强有力的动力之一，其革命性的发展不仅极大地影响着人们生活、学习和工作的方式，而且正在迅速地成为世界经济增长的重要动力。信息技术所带来的经济与社会变革的实质，就是帮助个人和社会更好地使用知识和智慧，从而使人类充分发挥其潜力，实现其理想。近年来，信息技术快速发展，数字鸿沟也日益加大。它不仅存在于国与国之间，也体现在一些国家内部各阶层人员之间。在获取和使用信息技术方面，城乡之间、贫富之间、受过教育的人和文盲之间，男女之间，残障人与非残障人之间，老年和青壮年之间，也有着显著的差异。数字鸿沟的本质是信息资源获取和利用的不平等，这一性质正阻碍着建立公平和充满活力的信息社会。信息无障碍就是要消除数字鸿沟，无论是健全人还是残疾人，无论是年轻人还是老年人，在任何情况下都能平等地、方便地、无障碍地获取信息，利用信息。它将是 21 世纪残疾人服务中最重要课题之一。

要想达到信息无障碍，就必须对信息技术产品进行无障碍测试，使其达到信息无障碍的要求和规范。

11.1.2 无障碍测试的定义

无障碍测试软件产品、网站或其他产品，目的是看测试的产品能否被多种用户特别是残疾人群和特定人群使用，是否支持盲人使用的屏幕阅读器，这其中也包含了正常人在某些时候发生暂时性障碍的情况下能否正常使用，以及是否支持全键盘操作等。

无障碍测试可以使用普通测试中的手工和自动测试方法，也可以采用调查访问等方式。例如让一群有代表性的用户(或模拟这些用户)，主要是盲人等残疾人用户，在他们尝试对产品进行典型的操作时，由观察员和开发人员在一旁观察、聆听，做记录，并根据这些记录的结果，对产品进行关于可用性的改进。该产品可能是一个网站、软件或其他任何产品。测试可以是早期的纸上原型测试，也可以是后期成品的测试。

为了让所有人都可以便利地使用软件或网站，需要改进一些以前很少考虑的问题。例如，在网页中不可调整字体大小的小字体文档，会使一些视力有限的用户无法阅读文档

内容；细小的导航按钮只有很小的单击范围，不方便一些身体机能有障碍的用户使用；闪光或闪烁的页面甚至可能对患有癫痫症的用户产生生命危险。以上问题都可以归结为无障碍测试需要检查出并改正的问题。

无障碍的软件或网站的内容，应该可以被以下残障人浏览和使用：

- 由于视觉障碍（弱视或色盲）或视觉损失等无法识别屏幕中信息或看不清屏幕的人。
- 由于生理障碍而无法控制鼠标或键盘的人。
- 由于听觉障碍或听觉损失而无法接收软件或网络语音信息的人。
- 由于认知障碍（失语症、学习障碍或记忆功能障碍）而无法理解软件或网页的内容的人。
- 由于文化障碍（阅读能力差或阅读非母语内容）无法完全理解软件或网页的所有内容的人。

实现了无障碍的网站的受益人群不仅包含上述人群，也能让以下人群受益：

- 善于和不善于交流的人群。
- 老年人和新手，他们通常都是计算机盲。
- 使用老旧计算机的人群（无法运行最新的程序）。
- 使用“非标准”设备（PDA 或移动电话）的人群。
- 处于受限环境的人群。
- 由于环境影响而产生临时障碍的人群。

11.1.3 了解无障碍测试

为了让读者对无障碍测试有个初步的认识和了解，这里介绍一下 Web 无障碍测试例子。

1. 页面样式和字体选择

一般而言，对于残障人士，尤其是视力不好的用户，大的字体、间隔分明的布局体系有利于他们阅读 Web 网页。他们也通常会利用浏览器自带的放大缩小字体的功能来为自己设置最为合适的字体。不同的浏览器缩放字体的方法不尽相同，在 Firefox 浏览器中可以使用 Ctrl＋＋快捷键来放大字体，Ctrl＋－快捷键来缩小，而 IE 浏览器则可以使用菜单上的字体缩放调整。

字体的使用应该尽量使用比较规范的印刷体字体，而不要使用一些不常见的手写体形式。字体的大小可以在层叠样式表 CSS 中定义百分比，或者以 em 等为单位设置字体大小，从而支持动态缩放。这样的字体单位属于相对单位，各种浏览器都能较好地支持。同样也可以在 JavaScript 中为页面上的某些节点方便地设置字体大小，这主要用于一些 Ajax 的应用程序中。

如果不使用 CSS，而去使用 JavaScript 定义字体大小是不常见的，但是对于一些视力

有问题的用户应用不熟悉的浏览器，可以在网页的显眼位置设置增大、缩小文字的按钮，用户通过单击这两个按钮，而不需要掌握浏览器伸缩字体的方式，就能够放大、缩小网页上的字，很显然，这种按钮的背后就是上面使用的 JavaScript 在起作用。

而更加人性化的完备的解决方案则是多样式表的提供。例如说专门为视力弱小的人群提供无障碍的 CSS，用户可以自主的选择这种样式为整个网页“换肤”，并且，浏览器通过 Cookies 记录下来用户的偏好样式，以后每当该用户重新访问该网页的时候，就将该用户经常使用的无障碍 CSS 自动呈现。

2. 页面导航和键盘支持

用户在浏览网页的时候，经常需要与网页进行一系列的交互操作，例如单击链接、输入表单信息、单击按钮等。大部分的用户相当依赖于鼠标来进行这些工作，鼠标能够精确定位页面上任意一处地方，并且通过单击左键、中键、右键等来触发相应的操作。但是这些鼠标的操作对于某些残障人士来说就很困难，他们很大程度上更加依赖于键盘的操作，尤其是 Tab 键、Enter 键、方向键等键值。其中最重要的可能是 Tab 键了，Tab 键主要用来进行页面元素的导航。

3. 使用 Tab 键导航

用户可以使用 Tab 键(或者 Shift＋Tab 键)来定位到页面上的元素，获得该节点的焦点，继而可以使用 Enter 键或者 Space 键来触发点击事件。这是最为普遍常见的键盘导航的应用。

在一个 HTML 网页中，一般意义上可以获取焦点的元素，也就是能够被 Tab 键访问的元素，主要是链接＜a＞标记、表单输入域(＜input＞，＜textarea＞标记等)、按钮等需要用户进行交互的元素。如果网页中不特别规定元素的导航顺序，用户使用 Tab 键，将依次按页面的顺序从左至右、由上而下(某些中东国家使用从右至左的文字顺序)的访问，Shift＋Tab 键则是倒序的访问。

当然软件开发人员可以设定元素的 TabIndex 属性，来自行规定访问的顺序。例如说 a、b、c 三个元素，tabIndex 依次是 2、3、1。那么使用 Tab 键导航将依次访问到 c、a、b。这些都是普通 Web 程序，为了增强无障碍性的常用方法。

而针对 Ajax 形式的 Web 应用程序，由于内容的动态性，访问的次序以及焦点的设置都有可能发生变化。例如用户通过点击某个按钮激活了一个利用 JavaScript 技术模拟的模态的对话框 Dialog，那么这个时候用户的页面焦点则应该被设定到这个 Dialog 中，并且用户再使用 Tab 键导航，则不应该跳出这个模拟的模态 Dialog 的范围，直到 Dialog 被销毁。

很显然，只是简单地使用 HTML、语言规范的 TabIndex 属性是做不到这一点的，必须拦截用户的键盘事件，来判断用户按的键值，并且根据不同的键值做出不一样的响应。

不幸的是，不同的浏览器对于用户按键的捕捉，键值的属性等都各有差异，很难做到支持各种主流的浏览器，不过 JavaScript 针对键盘事件的处理，尤其是各种键值在浏览器

中的属性不一样、事件机制也有差异的情况下，进行了规范化，屏蔽了浏览器之间的差异性，用户可以在同一个 API 下面进行编程，大大改善了方便度。

4. 使用其他键值进行辅助、快捷的操作

如果使用过 Google Reader，一个在线的 RSS 内容阅读 Web 应用程序，可能知道使用 j 键来打开下一条 RSS 的 Feed 内容，或者使用 k 键来打开上一条内容等。这些键值不同于 Tab 键、Enter 键等，HTML 网页中具有某些默认行为的键值，它们是自定义的，换言之，不同的 Web 站点可能有不同的操作规范，但是不管怎样，它们提供了除鼠标操作之外的另外一种方便快捷的途径，大大提高了 Web 应用程序的易用性。

而对这些键盘操作的支持，同样需要利用 JavaScript 捕获键盘事件，并针对不同的键值进行不同的操作，实际上，跟上面的例子基本上一样，只需要再加上对其他键值的判断和对应的处理方法，就可以基本上实现了。以下就是在网页中使用键盘捕获并进行处理的函数。

```
function onkey(evt){
        var key=evt.keyCode;
        if(key==dojo.keys.SHIFT_TAB)
        {
                // navigate to prev item
        }
        else if(key==dojo.keys.TAB)
        {
                // navigate to next item
        }
        …
        else if(key==42)
        {
                // user pressed "j"
                // do something.
        }
}
dojo.connect(document, "onkeydown", "onKeyPress");
```

整个函数实际上很简单，网站的无障碍性实际上不需要开发者编写很烦琐的代码，每增加一点支持，网站就更具有亲和力。

5. 读屏软件的支持

读屏软件是一种专门为视力低下甚至完全眼盲的人开发的辅助性软件，它能跟踪当前页面上的焦点，并且通过语音发声的形式告诉用户当前是什么信息。目前较为流行、用户范围较广的软件主要是 JAWS、Windows-Eyes 等。

(1) 支持基本的读屏操作

目前，读屏软件阅读网页的能力主要来自网页中 HTML 元素给出的相应信息。例

如链接 a 标记的 title 信息，图片的 alt 替代文字说明等。这就要求在编写无障碍网页的时候，需要为这些相关的信息元素加上这些辅助的信息，尽管这些信息在大多数时候看起来好像是毫无用处的，但是正是有了这样的信息，读屏软件才能帮助这些弱视的群体辨别当前信息内容。

对于 Ajax 形式的 Web 程序，这个尤为重要，因为内容可能时时都在发生变化。对于动态生成的内容，也不应该为读屏软件的工作设置障碍，主动设置好这些辅助信息是一个好习惯。

（2）支持 ARIA

基本的读屏支持是通过读屏软件阅读 HTML 元素给出的 title、label、标准表单元素的选中（checked，unchecked）信息来实现的，而目前因特网应用越来越多地使用一些自定义的、模拟桌面 GUI 程序的 Web widget 小组件，来增强 Web 应用程序的交互。由于这些带有某种模拟性质的页面节点，读屏软件是不能了解其真正的含义的，它只能去阅读其中的一些诸如 title、label 等信息，很难让弱视的用户获得和普通用户一致的感受。

ARIA 是为了解决类似于上述这种问题的标准协议，它的全称是 W3C Web Accessibility Initiative Accessible Rich Internet Applications（WAI-ARIA）Roadmap，应该说目前还正在完善之中。IE 这样的浏览器还是不支持，Firefox 浏览器支持得比较好。它定义了一系列附加的属性（角色 Roles 和状态 State）来支持描述页面上的这类事物。通过为 HTML 的 DOM 节点，例如一个 div 标记设置角色以及状态的信息，然后通过读屏软件的支持，来阅读出当前所访问的是什么东西，并且它处于什么样的状态，从而达到无障碍。例如对话框 Dialog 通过设置在 div 上的角色信息，用户能够知道当前访问的是一个对话框，并且能通过设置在 div 上的状态信息，得知这个对话框目前的状态等。

6. 支持高对比度的显示模式

某些弱视群体例如色盲患者，经常需要调整操作系统的颜色设置来达到高对比度的效果。而在 Web 程序中，由于大量使用了图片元素传达信息，这些用户可能对识别这些图片颜色感到困难，所以也可能选择关闭这些图片的显示，而仅依赖于图片的提示说明文字（例如说图片＜image＞标记的 alt 属性）。但是在 Web 程序中，由于图片之前占据了页面中的一些位置，禁用这些图片将导致网页信息不完整，用户得不到相关的完整信息。

这时可使用 JavaScript 来侦测浏览器是否处于这个高对比度的无障碍模式，而进行不同的输出（针对一般模式，输出背景图片，而针对高对比度的无障碍模式，使用可代替的文字占位符等代替图片），保留了信息的完整性，从而避免了这些弱视用户的信息获取的不完整。

以上只是介绍了无障碍的部分内容，以上提到的文字大小、布局样式、读屏发声、高对比等的支持是一个具备无障碍性的 Web 程序的主要部分。虽然大多数人不需要这样的信息，但是为了让用户能更加广泛、多元化，更公平地参与网络带来的信息世界，无障碍的重要性是不言而喻的。

11.2 无障碍标准和规范

11.2.1 软件无障碍

软件无障碍的参考标准很多，美国就出台了关于无障碍的实施法律——美国 508 条款。现在国外很多公司都是依据这些标准来制定无障碍测试流程，保证自身公司产品满足无障碍的需要。

1. 美国 508 条款

美国 508 条款（Section 508）是对 1973 年的《康复法案》的修正案，它开创了全球信息无障碍立法工作的先河，它是一部联邦法律，规定了所有由联邦政府发展、取得、维持或使用的电子和信息技术都必须让残疾人无障碍。

美国 508 条款特别强调电子信息资源要能够被残疾人群访问，并成立了一个建筑和运输障碍委员会（the Architecture and Transportation Barriers Board，简称 Access Board），要求其制定相关无障碍性标准。其中的电子与信息技术无障碍最终标准，对软件无障碍做出了具体的规定，该标准于 2000 年 12 月 21 日提出。

2. 有效的色彩对比标准

这个标准是 Lighthouse International 组织制定的，该组织于 1905 年创立于纽约市。它是全球领先的非营利性组织，致力于保护视力的研究和帮助不同年龄的人克服失去视力的困难。该组织通过临床服务、教育、研究和鼓励等方法，帮助那些弱视者和盲人过上安全和独立的生活，其网址为 www. lighthouse. org/color_contrast. htm、www. lighthouse. org/color contrast. htm。

该文档为制定有效的色彩选择提供了指南，指南提出了 3 条基本原则，目的是为所有人作出有效的颜色选择。因此主要内容是解说 3 种颜色的感知属性，包括色调、亮度和饱和度。

3. 文本清晰易读标准

这个标准是 Lighthouse International 组织制定的，该标准文档为使文本清晰易读提供了基本指南，其网址见 http://www. lighthouse. org/print_leg. htm。

4. EITACC 桌面软件标准

该桌面软件标准由 EITAAC（美国电子与信息技术无障碍顾问委员会）制定，由美国教育部于 1997 年 3 月 6 日出版，版本为 1. 1，该标准见 http://www. trace. wisc. edu/docs/eitacc_desktop_software_standards/desktop_software_standards. htm。该标准为桌面软件定义了无障碍标准，要求教育部门员工使用和采购软件时必须符合此标准，并鼓

励根据此标准开发软件和教育技术，以达到无障碍。

此外该网站还提供了信息技术无障碍需求说明，包括了设计无障碍软件的详细指导。

5. MAGpie 标准

MAGpie 标准是由国家媒体无障碍中心（The Carl and Ruth Shapiro Family National Center for Accessible Media，NCAM）开发的，这个研究中心 1993 年成立，致力于研究和开发设备，以帮助那些在家中、学校、工作场所和社会的残疾人士解决媒体和信息技术的无障碍问题，例如给视频添加字幕等。美国 AOL、Google、微软等都是其战略合作伙伴。

MAGpie 标准是该研究中心 Web 无障碍项目衍生的标准，该项目从 1996 年开始持续到现在，已卓有成效，例如产生了视频无障碍产生器、Web 访问语法等。MAGpie 可以帮助多媒体开发者使他们的作品达到无障碍，NCAM 已经开发了配套软件 Media Access Generator，最新版本为 1.0，这个软件能够为各种多媒体创建字幕和音频描述。

其网址见 http://main.wgbh.org/wgbh/pages/ncam/webaccess/magpie/index.html，该网站还提供了其他无障碍说明，例如可提供媒体格式说明，能够加载字幕的工具和播放器，还有视频描述的相关信息。

6. KDE 用户界面指南

KDE 是 K 桌面环境（K Desktop Environment）的缩写，一种著名的运行于 Linux、UNIX 以及 FreeBSD 等操作系统上面的自由图形工作环境。

以前在 UNIX/X11 下创建应用程序，是一个非常困难并且单调乏味的过程。KDE 认识到了在一个计算平台上，平台和对于这个特定平台用户可用的一流应用程序的集合是同等重要的。从这些观点出发，KDE 项目已经开发了一流的复合文档应用程序框架，实现了最先进的框架技术，并且因此把它自己直接置身于和诸如微软的 MFC/COM/ActiveX 技术等流行开发框架相竞争的位置。用户界面指南就是在此背景下提出来的，目的是提供一个美观的桌面，所有的 KDE 应用程序都具有统一的视觉观感，如标准化的菜单、工具栏、键盘绑定、颜色样式，并得到国际化支持等。这些是无障碍标准的很重要的因素。

KDE 用户界面指南是一个在线文档，其中包括 KDE 应用程序在 UNIX 环境下的用户界面指南。其资料见 http://developer.kde.org/documentation/standards/kde/style/basics/index.html。

7. Motif 与 CDE 2.1 风格指南

Motif 最初是由 OSF（开放基金协会）开发的一个工业标准的 GUI（图形用户接口）。Motif 最先实现并运行于支持 X 窗口系统上，它是 UNIX 系统的主要用户接口。目前已经应用于 200 多种硬件和软件平台。Motif GUI Toolkit 推动了网络环境下的应用开发，各种机器包括便携机、PC、工作站、超级计算机都得益于 Motif 环境下的应用程序一致的行为。用 Motif GUI 开发的应用软件，具有高度的可移植性、可交互性及可伸缩性。该

指南提供设计和开发新软件的开发者一个行为规范的框架，这个框架是与 Motif and Common Desktop Environment (CDE) 用户接口保持一致的。这种行为是在拟定目前多种行为模型的共同原理的情况下建立的。Motif and CDE 2.1 风格指南是一个 Online documentation which covers design issues for the Motif and Common Desktop Environment (CDE) user interface 在线文档，主要介绍 Motif and Common Desktop Environment (CDE)用户界面的设计问题，其网址为：http://post.doit.wisc.edu/library/techlib/manuals/adoclib/motif/motifsg/toc.htm。

8. IBM 信息技术标准

1996 年，EITAAC 副主席兼 IBM 公司美国无障碍中心研究员 Jim Thater 博士主持制定了 IBM 无障碍使用指南，在 IBM 公司内部使用。在 IBM 信息技术标准中，对外公开的有 IBM Java 无障碍检验表，为开发和设计无障碍的 Java 应用程序提供了详细的基本原理、技术和测试方法，见 http://www.ibm.com/able/accessjava.htm。IBM Lotus Notes 无障碍检验表即用于在 Notes 客户端或 Web 上进行设置的 Notes 应用程序等检测表。具体可以访问 IBM Human Ability and Accessibility Center 网站 http://www.03.ibm.com/able/index.html。

9. 微软无障碍标准

微软公司其实很重视无障碍技术研发，常用的 Windows 操作系统控制面板里面有一个选项叫辅助功能，这就是无障碍技术的一个体现。该功能能够提供字体放大、高对比度等功能。

为了使微软系列产品达到无障碍，也为了帮助开发人员开发的 Windows 应用程序也能达到无障碍，微软公司出台了以下无障碍标准和技术。

① Active Accessibility：1997 年开发，将提供给开发人员用于对程序和操作系统与辅助功能共同工作的方法加以改善，以帮助软件开发人员使其程序与辅助功能更兼容，使辅助功能更可靠。目前已用于微软的操作系统和程序。Active Accessibility 为开发人员提供了一种标准方法，该方法就是将用户界面元素信息传递给辅助功能选项。这些信息包括对象类型、对象名称、对象位置、对象的当前状态等，还有其他诸如导航、Windows 事件通知等信息。此标准还令使用 Active Accessibility 的开发人员在设计其程序用户界面时，有更大的灵活性。了解了辅助功能需要什么，软件开发人员能够在不牺牲兼容性的情况下，更自由地进行创新。另一方面，帮助辅助功能开发人员树立信心，他们创建的产品会与使用 Active Accessibility 的程序很好地协调工作。具体网址为 http://www.msdn.microsoft.com/library/default.asp。

② 无障碍软件设计指南：该文档为开发和设计无障碍 Windows 应用程序提供了详细的指导，讨论了如何让计算机对残疾人是无障碍的，以及如何设计和开发适用于残疾人的软件。遵循该指南，有助于消除用户使用软件的障碍。网址为 http://msdn.microsoft.com/library/default.asp? url=/nhp/Default.asp? contentid=28000544。

③ 用户交互设计指南：这本指南在 1995 年出版以来，还在继续发展。其中新的版本

已用于 Windows Vista 和 Internet Explorer 7，开发人员可利用该指南理解软件界面和优化设计工作。它包含强制性的规格说明和实用的设计建议，从而为开发人员提供了参考。

10. Macintosh 人机交互指南

该指南由 Apple 公司推出，适用于设计和开发 Macintosh 产品的人员。它可以帮助设计师、人机交互专业人员或工程师设计和创建适用于苹果公司产品。其缺点是要求开发人员熟悉 Macintosh 的概念和术语以及曾经使用过 Macintosh 计算机和一些 Macintosh 软件。其指南文档见 http://www.devworld.apple.com/techpubs/mac/HIGuidelines/HIGuidelines-2.html，这是一个在线文档，其中包括 Macintosh 应用程序的设计问题。

11. 其他标准

像 PDF 格式文件、real 媒体文件等都会经常用到，但这些文件是有障碍的，常见的问题就是视力障碍用户无法用屏幕阅读器来阅读这些文件。为了解决这些问题，相关公司提出了文件无障碍标准。

① PDF 设计无障碍文档：PDF（Portable Document Format）文件格式是电子发行文档的事实上的标准，为了使 PDF 文档达到无障碍要求，Adobe 公司提出了 PDF 设计无障碍文档，主要内容有 Adobe PDF 文件无障碍解决方案、优化 Adobe PDF 文件并实现无障碍的技术等关于设计无障碍 PDF 文件的资料。其网址为 http://www.adobe.com/products/acrobat/pdfs/pdfaccess.pdf（link resides outside of ibm.com）.acrobat/pdfs/pdfaccess.pdf。

② real 媒体无障碍文档：为了使 real 媒体文件，特别是适合在网上播放的流媒体文件达到无障碍，Real 公司提出了 7 条原则，允许用户创建字幕、打开字幕等。其网址是 http://www.real.com。

无障碍标准案例如表 11-1 所示。

11.2.2 Web 无障碍

Web 标准对于互联网来说是非常重要的，理论上可参照前面所述软件无障碍标准和规范，但又有自己的特点。IT 宪章指出，互联网在不断发展，它已经不是一种简单的静态信息和平台了，而是一种全球性互动式的环境。新技术的不断发展，网站使用的日新月异，各种软件工具的不断出现，意味着用户的数量和价值都在增加。信息无障碍就是为用户参与到这个新环境中来，拥有平等的使用权利，并跟其他人一样享受到科技进步和社会发展的成果奠定了坚实的基础。因此万维网联盟（World Wide Web Consortium，W3C）专门成立了 Web 无障碍组织（Web Accessibility Initiative），致力于制定和推广 Web 无障碍标准，其中 WCAG 1.0 是影响最大、意义最深远的 Web 无障碍标准。世界各国制定信息无障碍标准和法律时都参照了 WCAG 1.0。

表 11-1　IBM 软件无障碍检测表

1	键盘访问	是否计划 N/A	备注
1.1	为所有操作提供可替代的键盘操作		
1.2	不影响内置在操作系统里面的键盘无障碍性的特性		
2	对象的信息	是否计划 N/A	备注
2.1	当输入焦点改变时，提供一个视觉焦点指示器，在交互式对象之间移动该焦点指示器必须以编程的方式体现无障碍技术		
2.2	提供涉及语义方面的关于用户界面对象的信息。当一个图像表示一个程序单元时，该图像所传达的信息必须能够在文本中可得		
2.3	将标签与控件、对象、图标和图像联合起来。如果图像是用来识别方案内容的，那么它所包含的含义必须始终与应用程序保持一致		
2.4	当使用电子表格时，在所有的指示与提示的帮助下，表格将允许人们使用无障碍技术查阅资料、域元素和所必须完成的操作，并提交表格		
3	声音和多媒体	是否计划 N/A	备注
3.1	提供一个选项，显示所有音频警报的视觉提示		
3.2	为有现实意义的音频和视频提供无障碍的替代物		
3.3	提供一个选项调节音量		
4	显示	是否计划 N/A	备注
4.1	通过支持与无障碍技术进行交互的标准的系统功能，调用或一个 API（应用程序编程接口）的方式提供文本		
4.2	使用颜色作为加强，但不是作为唯一传达信息的途径或者表明一种操作		
4.3	对所有用户界面控件和客户方面的内容都支持系统高对比度设置		
4.4	当支持颜色个性化时，提供多种颜色的选择能生产一系列的对比水平		
4.5	为所有用户界面控件继承系统设置字体、大小和颜色的功能		
4.6	提供一个以非动画演示模式显示动画的选项		
5	时间调整	是否计划 N/A	备注
5.1	提供一个选项，以调整按时指示或允许持续指示的响应反应		
5.2	不要使用闪光或者闪烁的文字、对象，或其他闪光或闪烁频率大于 2Hz 和低于 55Hz 的元素		
6	无障碍验证	是否计划 N/A	备注
6.1	使用可用的工具进行无障碍测试		

1. Web 无障碍关注内容

互联网协会的主席，同时也是互联网的发明者 Tim Berners-Lee 曾经说过，"网页的威力在于其广泛性，每一个人，无论障碍与否皆能使用，才是最关键的要素。"尽管目前的互联网充满了大量的视觉元素，制造了眼花缭乱的互动效果，但是并非每个人都能享受到这种感官刺激。有许多互联网用户在身体上存在障碍，他们因为全盲而依赖点字显示器或屏幕阅读软件来浏览网页，或者因为弱视而必须把网页文字调到非常大才能看清楚，或因为色盲而无法分辨链接文字的色彩变化，或者因为肢体障碍而无法使用鼠标。对于这些用户来说，无论多么精彩的网页，都将变得难以阅读、难以操控、难以理解，甚至有些用户恐怕连网页到底是什么都无从得知。以下无障碍标准和规范所关注的正是这些特殊用户的诉求，它的相关标准确保了每一个用户在各种情况下都能顺利获得信息。

2. WCAG 1.0 介绍

WCAG 1.0 即 Web 内容无障碍指南 1.0 版(Web Content Accessibility Guidelines 1.0)是由万维网联盟的 Web 无障碍组织制定并发布的。在 1999 年 5 月，WCAG 1.0 就已经成为互联网协会推荐的官方标准，并得到广泛认同。

WCAG 1.0 是考虑各类残疾人群在访问网页内容时的特点而制定的与之相对应的技术标准，它对网页设计者提出了具体的网页内容无障碍要求。WCAG 1.0 讨论了各类无障碍性问题，并提供了无障碍设计解决方案。每一条都针对特定的场合、特定的人群可能产生的问题来解答。WCAG 1.0 针对 Web 内容无障碍设计提出 14 条指导原则，每条都包括序号、综述、导航链接、基本原理与受益人群、检验点定义(Checkpoint Definition)列表等。检验点定义的作用主要是解释每条指导原则在网页设计时具体的应用条件和应用方法，还包括序号、综述、优先级别、资料信息以及技术文档的链接。其中最突出的是优先级别的设置，它按照检验点对网页无障碍性的重要程度，赋予每个检验点一个优先级别。WCAG 1.0 已经成为国际社会最著名的 Web 无障碍技术标准之一，并且还出现一些依据 WCAG 1.0 的标准来评价任何一个网页无障碍情况的工具软件，如 Babby Online Portal。

(1) WCAG 1.0 所倡导的理念

作为国际社会公认的 Web 内容无障碍技术标准，WCAG 1. 0 倡导的理念主要有两个：保证网页内容的良好呈现，让网页内容易于理解和具有导航功能。

① 保证网页内容的良好呈现。

为了符合 WCAG 1. 0 所要求的各类技术标准，网页内容开发者必须能够保证网页内容的良好呈现。即使用户出现多种问题，包括身体上的、感觉上的和认知上的残疾，工作上的限制以及技术壁垒方面的问题，网页内容仍旧要保证一定程度的无障碍性。保证网页内容的良好呈现必须具备以下 4 个关键要素：

第一，把网页内容的结构与呈现形式分离开来。网页内容的结构是指网页内容在逻辑上是如何组织起来的，例如，通过章节、引言、目录等。呈现形式是指网页内容按照哪种媒体表现形式呈现出来，例如，打印文本、纯文本、二维图像、语音合成、盲文等。

第二，提供文本，包括替代文字说明（Text Equivalents）。因为只有文本可以被大多数的浏览设备识别，并且对几乎所有的用户都具有无障碍性。

第三，为各类用户创建信息等价物，包括盲人和聋人，使他们能够通过其他形式获得视觉和听觉信息，例如音频、视频或其他感觉辅助功能。当然，这并不意味着要在网页内容的所有位置插入音频信息以保证盲人获取有效的信息，他们只要使用屏幕阅读器（Screen Reader）就可以了。聋人也不外如此，只要使用数据手套、位置跟踪器就可以实现了。

第四，网页内容的呈现应该不受硬件设备的限制，页面应该在任何情况下都可以正常工作。例如，在没有鼠标、小屏幕、低分辨率、单色屏幕甚至无屏幕、只有语音或文字输出的条件下良好呈现网页内容。

WCAG 1.0 的第一至第十一条指导原则就是依据该理念进行制定的。

② 让网页内容易于理解和具有导航功能。

网页内容开发者必须保证网页内容易于理解和具有一定的导航功能，其中不仅要求语言简单明了和通俗易懂，还要求为页面之间的切换提供言简意赅的导航功能。在页面中呈现导航工具和索引信息，可以使网页内容的无障碍性和易用性达到最大化。并不是所有的用户都可以使用一些视觉化的网页导航信息，如网页地图、滚动条窗口、框架页及页面映射等。很多特殊用户也许无法获得网页上下文的信息，他们只能获得网页一小部分的信息，或者只能通过逐词阅读获得信息（通过语言合成功能、点字显示或对屏幕的局部放大）。失去导航信息，用户就根本无法理解一些大型的表格、列表和菜单等。因此，导航功能是无障碍性网页的一个基本要求。

WCAG 1.0 的第十二至第十四条指导原则就是依据该理念进行制定的，具体指导原则可参见附录 C。

（2）WCAG 1.0 的重要作用和发展前景

WCAG 1.0 的内容没有采用一些死板的符号规则来表示，而是采用十四条明晰且提供具体操作规范的指导原则来表示，目的就是为了避免网页开发者敷衍了事。不少网页开发者并不看好无障碍网页，以为无障碍网页必定是枯燥无趣且呆板丑陋的。然而事实并非如此。WCAG 1.0 的指导原则并不是要限制网页艺术家想象力和艺术才能的发挥，而是希望在作品完工之前，能再多花点心思加以修补和解释不同媒体形式的信息，让每个细节都尽善尽美，让更多的人能够领略作品的美好。

WCAG 1.0 从制定、实施到现在已经快 8 年了，WCAG 2.0 的技术工作草案已经制定，且处于完善之中。与 WCAG 1.0 相比，WCAG 2.0 所包含的网络技术，包括 Java、PDF、Flash 都将是完全技术性和独立的。另外，W3C 组织建立了一个无障碍标准的协调小组，准备组建一个标准化的储存库，届时所有国家都可以参照这个储存库的标准，制定自己国家的无障碍标准，推动信息无障碍事业的发展。

3. WCAG 2.0 解读

与 WCAG 1.0 相比，WCAG 2.0 的变化是巨大的，它以完全不同于 WCAG 1.0 的方式组织内容。WCAG 1.0 由一整套指导方针组成，每条指导方针包含一系列的检查点，

这些检查点(www.w3.org/TR/WCAG10/full-checklist.html)解释了如何在网络开发中应用指导方针,按照优先级1、2、3排列。WCAG 2.0与此完全不同,它以适用原则、仿真和成功标准来组织内容。另外,成功标准没有按优先级排序,取而代之的是分成不同的等级。

WCAG 2.0新规范与WCAG 1.0相比,拥有更多的技术独立性,并且以更加普遍的方式来论及基于互联网的无障碍问题。WAI努力使新指南能够适应这些变化和发展的技术,并且能够在将来用于网络开发的模式和技术中检验网页的无障碍性。

在指南和原则更普遍化的同时,WCAG 2.0也提供了丰富的材料用于理解如何达成一致。这些材料是具体而有用的。WCAG 2.0仍然处于发展中,因此它也鼓励那些参与到无障碍领域的人对此做出贡献。

WCAG 2.0指出无障碍上网首先要保证网页可以被任何人直接或借助辅助工具访问到,要求网页设计做到以下几点:

① 网页内容必须是可感知的;

② 网页内容中的界面组件必须是可操作的;

③ 网页内容和控件必须是可理解的;

④ 网页内容必须足够健壮,能够与当前及未来的用户代理(包括辅助技术)协同工作。

这4项原则为任何人访问和使用Web内容奠定了必要的基础,并为提高残障及特定人群感知、操作和理解Web内容的能力提供支持。每项原则之下都有许多特定的指南来实现该原则。而在每一项特定的指南之下又有若干合格标准来评估是否符合该特定指南。每项原则之下的指南的合格标准被划分为三个等级:分别为能够实现最低级别、可以实现和强化级别的无障碍访问,并可以合理地应用到所有Web内容中以及可以实现增强的无障碍访问,而不需要应用到所有Web内容中。

这些原则、指南以及合格标准表达了解决无障碍访问问题和需求的各种概念。而无论采用什么技术,它们并非针对HTML、XML或任何其他技术。这种方式有利于把标准的规定应用于多种场合和技术中,包括那些目前尚不存在的场合和技术。

前面已经说明美国的508条款是对1973年的《康复法案》的修正案,其中特别强调电子信息资源要能够被残疾人群访问,当时跟WCAG 1.0并没有什么关系。但在WAI发布WCAG 1.0之后,美国于2000年12月也重新修订并公布了508条款,该条款就是根据WCAG制定网站应该满足无障碍性的要求,要求政府和学术性的网站必须满足无障碍性的要求。该法案于2001年6月生效,在此之后开发的网站必须遵守该法案,尽管该法案对在这之前建设的网站没有作相应的要求,但是美国很多网站都自觉地按照该法案去修改自己的网站。

(1) 508条款整体目标

在制定508条款时,制定者规定了508条款的如下整体目标:

① 解决信息科技所带来的障碍问题;

② 促进残疾人群获得平等的机会;

③ 鼓励无障碍信息科技的发展。

(2) 508 条款鼓励发展的技术

① 计算机设备无障碍技术。

包括硬件、软件、接入设备，为因肢体残疾不能控制键盘及鼠标的残疾人设置特殊键盘，以便操作；为肢体移动不便的残疾人设计可以随意移动的计算机设备；为盲人设计屏幕阅读软件使用计算机，以便他们顺利获取软件的工作信息。另外，还可以在操作方式上设计键盘操作和声音控制视窗操作的解决方案。

② 网络资源无障碍技术。

在使用图形化、动画以及视频的表达方式时，加配文本说明，以便他们的读屏软件可以正确解析这些图形、动画和视频的声音，同时，计算机屏幕以大号字体显示，以便视力不好的人观看。设计将聋人所浏览的有声网页、语音聊天室、视频的声音转换成文字的支持等。

除此以外，还鼓励发展公共传媒无障碍技术、公共设施无障碍技术和通信设施无障碍技术等。

(3) 508 条款与 WCAG 的关系

508 条款关于网页内容无障碍的规则，从 2001 年 6 月开始正式实施，它要求所有的政府网站必须符合这些规则。任何为政府服务的签约公司，必须遵循这些规则来设计和开发网页，任何与政府有业务往来的公司以及接受政府资金资助的公司，都必须尽全力提供无障碍的网络服务。508 条款的完整文本以及 FAQ、留言板和培训信息都可在 www.section508.gov 上找到。

对于网页内容的无障碍性，联邦政府有 16 条规则。前 11 条是来自 WCAG 的检查点，后 5 条是专门针对 508 条款的，而不是 WCAG 的组成部分，这些规则可在 508 条款的 1194.22 条款中找到。

(4) 508 条款的影响

508 条款成为美国信息无障碍立法的典范，受到美国各级政府的重视，并得到严格的贯彻执行。在美国发生过多起因为违反这一规定而遭到处罚的案例，例如宾州政府门户网站对盲人存在障碍而引起诉讼，还有西南航空公司票务系统对残疾人存在障碍，美洲银行网站和自动提款机对残疾人存在障碍，美国在线的网页与屏幕阅读软件不兼容等都受到了相应处罚。

另外，508 条款目前已经成为国际性的网页内容无障碍的参照规范，许多国家都参照该规范建立起本国的信息无障碍标准。

Web 无障碍标准案例

IBM 网页无障碍检验表用于网站和网络应用程序。检验表的内容如下：

① 图像和动画。使用 alt="text"属性为图像提供可替代性文本。对于不是用于传递重要信息或传递冗余信息的图像使用 alt=""。

② 图像映射。使用客户端图像映射和图像映射热点的可替代性文本。如果需要服务端映射，就提供等价的文本链接。

③ 图和图表。概述每个图和图表的内容，或者使用 longdesc 属性来链接到描述文字和数据。

④ 多媒体。为重要的音频内容提供文字说明或抄本,为重要的视频内容提供抄本或音频描述。

⑤ 脚本。确保脚本的功能可以通过键盘操作。如果脚本影响到的内容不具有无障碍性,就提供一个可供选择的方法。

⑥ Applet 插件和非 HTML 内容。当一个 Applet 插件或其他应用程序需要出现时,为它们直接提供一个无障碍的链接,或者为那些不具有无障碍特性的提供替代内容。

⑦ 表单。使用辅助技术来确保表单无障碍。

⑧ 跳到主要内容。提供一种方法来跳过导航链接,直接到达页面的主要内容。

⑨ 框架。为每一个框架元素和框架页面提供标题,为每一个框架提供无障碍资源。

⑩ 表头。使用 TH 元素标记表头,在复杂的数据表格元素中使用标题属性。

⑪ 级联样式表。在不要求样式表的情况下,网页应该是易读取的。

⑫ 颜色和对比度。确保颜色所表达的信息在没有颜色的情况下,也能表达出来。

⑬ 闪烁、移动或颤动的内容。尽量避免内容闪烁、颤动或移动。

⑭ 同步响应。当需要同步响应时,提醒用户和给出充足的时间来指示需要更多的时间。

⑮ 纯文本页面。如果无障碍技术无法通过任何一种方法来应用,则提供一个纯文本页面,并附带等价的信息和功能。当原始页面改变时,就更新纯文本页面的内容。

⑯ 检验无障碍。使用有用的工具对无障碍特性进行检验。http://www.ibm.com/able/accessweb.html 中的文档为开发和设计无障碍的 Web 应用程序提供了详细的基本原理、技术和测试方法。

11.3 无障碍测试工具介绍

WAI 的评估和维修工具工作组列出了 3 个类别的超过 90 个软件工具:评估、修复和转换。这里将介绍以下 6 个商业 Web 无障碍测试工具,这些简短的说明都是来自其相应的网站。

1. Rational Policy Tester Accessibility Edition

来自 IBM 公司旗下 Watchfire 公司,整合了其前身 Bobby 和 Webxm 等软件功能,其 Bobby 是最知名的无障碍检测工具,因为它存在时间长,于 1996 年 9 月作为一个免费下载的工具由应用特殊技术中心(CAST)首次发布。2002 年 Bobby 被 Watchfire 公司收购。Bobby 在页对页的基础上通过一个网站测试,以查看是否符合多项无障碍要求,包括通过屏幕阅读器的可读性,并提供所有图像、动态元素,以及音频和视频显示的文本替代。Bobby 可以查看本地网页以及防火墙背后的网页。它可以完成超过 90 项无障碍检查。而其前身之一 Webxm 提供软件及服务,以识别、计算并排序无障碍优先顺序,以及企业网站的不符合规范的风险。

IBM Rational Policy Tester Accessibility Edition 有助于确保所有用户对 Web 站点的无障碍性。它帮助确定站点达到无障碍标准的级别，并在指示板和报告中显示结果。

2. Infocus

SSB BART 集团的 Infocus 桌面工具是第一个商业上的 Web 无障碍软件，在市场上仍处于领先地位。它提供了超过 115 个无障碍测试，其中包含所有的主要无障碍标准，具有较高的自动化水平。

3. LIFT Machine

Usablenet 公司的 LIFT Machine 是一个基于服务器的应用程序，它自动扫描内部和外部网站，提供超过 140 个质量无障碍和可用性的检查内容，然后为高级管理人员和独立内容创作者生成各种基于 Web 的报告。

4. Ramp Ascend

Deque 公司的 Ramp Ascend 提供了全部功能，包括为多媒体加入 SMIL 字幕，确保 Web 动画安全，并提供了全面的表格修补，即使是最复杂的 N 维图表。它包括了插件，这些插件可适用于 Macromedia 的 Dreamweaver、微软的 FrontPage 和 Mercury Interactive 的 TestDirector 8。

5. Webking

Parasoft 公司的 Webking 允许用户在浏览器中记录关键用户点击路径，然后它会自动配置和执行功能测试和回归测试，验证路径和网页内容，而忽略无关紧要的区别。Webking 的静态分析可以查找和定位不符合 508 条款无障碍规则的客户端代码，以及断开链接的页面、XML 问题和拼写错误。

11.4 无障碍测试实践

11.4.1 软件无障碍测试

软件作为一种特殊的 IT 产品，它的无障碍显得格外重要。在软件产品开发过程中，软件无障碍的测试是必不可少的一个环节。由于软件的特殊性，除了开发可以采用无障碍组件以外，软件无障碍测试没有专门的自动化测试工具来辅助测试，主要依靠手工测试来实现测试。一般要参照以下 3 个基本原则来进行软件无障碍测试，为了方便测试员的工作，各大公司还提出了软件无障碍测试列表，例如 IBM 软件无障碍检测表，测试人员可以按照列表进行测试。还有的公司进行用户测试一般是请有障碍的人群来试用软件，例如邀请视力障碍用户使用读屏软件来测试该软件。

1. 无障碍软件的 3 个基本原则

(1) 可选择的输入模式

输入方式包括键盘、鼠标、语音和可通过串行端口的辅助装置，首先要求能提供应用软件的所有细节和功能的全键盘操作(非鼠标操作)方法。

(2) 可选择的输出模式

输出方法包括显示器、语音和打印，要求能够为图标、图形和用户界面元素提供文本说明，并支持语音阅读。

(3) 一致性和灵活性

应用软件应能使用户的选择与系统的行为、颜色、字体大小和键盘设置等保持一致。

2. 软件测试技巧

以上软件无障碍测试三大基本原则只是提出了一个大的方向，最后还需要测试人员来实现。以下是无障碍测试人员常用到的测试技巧，通过这些测试来验证软件是否符合 3 个基本原则，或者像 IBM 软件无障碍检测表罗列的测试项目。

(1) 键盘操作

通过键盘的操作完成软件功能的实现。

① 软件应当同 Windows 常用快捷键一致，如特殊键命令，Alt+F4 键是关闭窗口，Ctrl+P 键是打印。

② 能够使用 Tab 键切换聚焦，使用 Enter 键进行相关的操作，使用 Shift+Tab 键向后切换聚焦。

③ 能够使用 F6 键或 Ctrl+Tab 键切换窗口。

④ 如果设置 SPI_GETKEYBOARDPREF 为 true 时，就显示了附加的键盘接口信息，这样用键盘操作更方便。

(2) 全键盘操作文档

提供全键盘操作方法的文档，包括进入菜单、键盘导航和快捷键。

① 提供键盘辅助的帮助文档。

② 提供键盘辅助的索引项。

③ 产品文档包含 keyboard access 部分，归纳所有辅助的功能点。

④ 提供与键盘操作等同的下拉菜单，标注好快捷键和加速键。

(3) 快捷键、记忆键和加速键的定义

给频繁使用的功能点定义快捷键、记忆键、加速键。

① 除了动态菜单，每个菜单项都应该有加速键。

② 可以参照 Windows 软件设计键盘用户接口。http://microsoft.com/enable 提供很多有价值的文档。

③ Windows 软件隐藏了一些用户接口信息，可以通过设置用户控制面板，显示出所有的附加的键盘用户接口信息。

(4) Tab 切换顺序

按钮和列表的切换顺序要符合逻辑。导航的顺序通常是从左到右,从上到下。

无障碍方面的建议如下:

不要改动和占用操作系统中已经存在的辅助功能点,如黏滞键和连续键等。

操作系统通常都包含一系列辅助选项,需要用户自定义设置键盘、显示、声音和鼠标等。

Windows 的键盘布置如表 11-2 所示。

表 11-2 键盘表

Keyboard Mappings	Mobility Access Feature
5 consecutive clicks of Shift key	On/Off for StickyKeys
Right Shift held down 8 seconds	On/Off FilterKeys
NumLock held down 5 seconds	On/Off ToggleKeys
Left－Alt＋Left－Shift＋PrintScreen	On/Off HighContrast
Left－Alt＋Left－Shift＋NumLock	On/Off MouseKeys

① Object focus:定义视觉焦点指示器,测试时注意菜单和对话框以及属性窗口等,是否有 Tab 键容易切换聚焦,并且聚焦是否清晰可见。

② Object information:对用户接口对象提供辅助操作信息。例如对盲人提供帮助作用的 screen reader 和 braille displays。screen reader 可以告诉用户当前聚焦的对象的属性和状态,测试的时候,打开 object inspector,把鼠标移到对象上,确认相关的信息是否正确。

③ Labels associated with controls or objects:控件的标记信息。测试的时候,打开 object inspector,观察控件的标注信息是否正确。

④ Positioning related objects:各项目按照逻辑或功能进行分组,有条理地安排顺序,并把重要的项目放到起始位置。测试的时候打开 screen reader,听各个控件和项目是否按照一定的逻辑顺序。

⑤ Option to display audio alerts:即使有声音警告的也要有视觉提示。

⑥ Visual display of information in video format:如果提供的重要信息是视频格式的,也要有提供同样信息的其他可达方法。测试的时候对软件输出的视频信息看是否可以用文本描述。

⑦ Disable sounds and adjust volume:提供关闭声音和调整音量的选项。

⑧ Displaying text:辅助功能的文本内容,包括文本输入脱字符号的位置和文本的属性。测试的时候打开 screen reader,确认是否能处理所有的文本。

⑨ Using color to convey information:确保所有信息在没有颜色的情况下都能呈现出来。要求对象要有文本描述,为了强调其重要性,最好有图形或字符的提示,如星号,也可以使用粗体或斜体区分文本。测试的时候,打印屏幕截图,确认所有信息都存在,对于 Windows 的应用软件,设置高对比度进行测试。

11.4.2 Web 无障碍测试

进行 Web 无障碍测试，必须首先明确：虽然无障碍测试以用户为本，但在产品成型后，通过用户反馈再来改进产品是不现实的。就计算机网站和软件产品而言，虽然改动起来要比其他实体类产品要容易得多，但在后期整体结构以及模块接口的修改所耗费的人力物力将远远高出项目的承受能力。

因此必须首先在设计阶段就进行无障碍功能的设计，并预留好无障碍方面的接口。熟悉 Web 无障碍测试所应用的工具和规范也很有必要。必须先了解 Web 无障碍有哪些要求，并在设计阶段就完成大部分的无障碍工作。后期的测试应该做的事是检查隐藏的问题以及代码编写中产生的一些错误，例如一些语法错误，虽然不会对整体呈现造成影响，但可能让辅助浏览工具失去作用。后期的测试不是应用无障碍检测工具和规范去慢慢检查整个网站，然后再一点一点修改。

在本章前面部分，已经对 Web 无障碍的核心标准以及检测工具进行了介绍。因此，Web 无障碍测试的内容，应更多地放在检测未发现的问题和改正这些问题上。

Web 无障碍测试可以由一个小组完成，由组长进行整体网站和网页结构的检查，由组员专门针对不同的功能进行检测。实际上，在网站的功能测试中，Web 无障碍的部分测试就能够很轻松地在功能测试时完成。我们要做的就是在功能测试没有问题后，更进一步检测这些功能是不是达到了无障碍的标准。例如说，如果网页某个部分能正确地显示一张图片，无障碍测试的内容就是根据标准，检查这张图片显示在浏览器中时，能否达到无障碍的相关标准，图片是否有完整可理解的替换文字等。这些工作做起来并不复杂，但最后对提高整个网站易用性的影响非常大。

Web 无障碍测试按工作进展可分为 3 个部分：制作者测试、全面测试和发布测试。这些与普通的 Web 测试基本相同，不过要做的工作更深入一步。

1. 制作者测试

包括美工测试页面、程序员测试功能。此阶段主要是对做好的功能进行即时性测试，因为此时代码和功能对程序员和美工而言是最熟悉的，检查出问题进行改正速度也最快。当然，这里所说的无障碍测试首先应建立在良好的工程设计基础之上，因此各方面的功能已经明确规定为符合整体项目的无障碍实现。在此阶段，可以改正大部分的影响浏览的问题。就 Web 代码而言，目前的代码测试工具只能测试纯 HTML 代码，因此，更多的工作需要手工检查。图片内容和声音内容的同步替换介绍文字是否能够正确地出现在页面中，动画图片是否闪动过快等，这些问题都需要在这一阶段进行检测和修正。

2. 全面测试

此阶段将是测试员以客户身份进行检查。主要检查的是整体页面的视觉效果，已经应用工具检查页面呈现的最终效果是否能够正常地支持无障碍功能。此阶段可以使用 ADESIGNER 工具来检查页面的视觉效果，以及使用 POLICY TESTER 等工具来检查

页面语法是否符合无障碍标准。同时，也应该将测试用户组引入，针对他们的反馈来进行发布前的最终修正。

3. 发布测试

这是网站发布到主服务器之后的测试。在正确运行后，要做的工作就是定期地进行无障碍的检查。因为网站内容和功能一直在增加，保证内容和功能都已经经过无障碍检测后再呈现出来，并且在修改完毕后，还要考虑增加部分对已有的页面产生的影响。例如说，早期的主页面已经经过良好设计并符合无障碍标准。此时加入某模块的内容(如新功能的简介)，就必须考虑到色调搭配等问题。

11.4.3　无障碍测试流程

没有一个作者愿意以非常失望的心态面对自己的作品。然而每一个网站都不能回避面对影响网站可用性的负面因素。如果首先利用有关无障碍标准和规范对自己的网站的无障碍性进行一个全面的审查，将会大大节省评估的时间。尽管静态测试是测试的第一步，但并不表明静态测试可以替代正式的无障碍测试。

静态测试首先检查软件或网站等其他产品是否启用了操作系统平台的无障碍辅助选项，如果没有，自己是否定义、编制和测试了自己的辅助选项。

其次利用网页或软件基础来检查，例如文字、图形、声音和超级连接等。像白色背景启用了浅黄色文字，就不醒目。网页或软件是否有正确的标题出现在浏览器或软件的标题栏中等。

常常被忽视的是替换文字，称为 Alt 文字。图 11-1 给出了一个 Alt 文字的例子。当用户把鼠标光标移到网页的图形上时，可以看到代表图形的弹出式说明。双目失明的用户可以借助 Alt 文字使用有声阅读程序，通过计算机的扬声器，听到解说和朗读的 Alt 文字。



图 11-1　Alt 例子

最后是必要的静态测试，例如代码检查、静态结构分析、代码质量度量等。

其中网页代码检查除了传统的检查以外，还附有如下特殊的内容：

- 网页中的图形、图像、图片按钮、Applet 等元素必须使用 Alt 文本来说明内容；
- 网页中的音频、视频、Flash 动画等应该加上相应的字母或配以 Alt 文本说明

内容；

- 网页要满足 HTML 4.0 规范，要有版本说明、标题、作者、关键词、描述等；
- 网页文件大小要适中，如果过大，用户在上网时兴趣很容易分散，下载时间过长令人难以忍耐；
- 网站导航机制要一致；
- 简约化的设计，这里要强调的是简约而不是简单。

网站其主要功能是使访问者获得信息、知识和技能，其次才是考虑网站的其他功能的实现，如网站的画面艺术、交互功能等。

① 第一步测试只是为了快速发现某网站存在的无障碍主要问题，测试并不深入和广泛。一般选择较少的用户进行测试或由网站的开发人员自己完成测试，可采取评测工具自动进行评测。

② 将清单提供给评估者，评估者越是独立于网站，他所提供的评价信息就越是有效。这些评估者一般要使用专门的无障碍测试工具来完成测试。

这一步主要是符合评测，按照已有的无障碍规范（如 W3C 的无障碍测试点）对网站进行一种定量的评测，可以得到网站对某个现有规范的符合程度。评测过程可采用工具评测、用户测试、专家评估 3 种方法结合进行。

③ 给评估者一些指导：通常评估者在出现问题的时候，会自然地认为是自己的问题，而不认为是网站的原因。这个时候，必须向他们解释清楚，不管他们认为问题的原因出在谁身上，需要他们将遇到的所有的问题都记录下来。

④ 正式的无障碍测试一般被作为可用性测试的内容。在这些测试里，实验者一般待在测试地点来观察和记录评估者的行为。但除非实验者是经过特殊训练的可用性专业人士，否则一直在测试地点会给评估者一个错觉，即正在进行的测试是针对他们的能力的。这种情况应该避免。

在 RUP 中实现无障碍功能，首先必须在需求分析以及设计阶段就将无障碍的需求考虑进去，并在测试中对其进行验证。目前一种可能的测试方法是在产品建模中添加标准用户角色和辅助用户角色，根据角色的需求在程序中添加检查点，通过 RUP 的自动化测试功能来对无障碍功能进行测试。并根据计划，在适当的时候邀请专家进行审查或用户来进行使用测试。

以上仅仅是初步完成了无障碍测试流程，按照 RUP 最佳实践经验，还要及时反馈给开发人员，进行反复循环迭代，直至软件或网站成型交付。

11.4.4 序列及交互化无障碍测试

在以往的无障碍测试中，会对照 IBM 公司的检查点逐一审核测试产品的各个页面和功能。这是一个复杂而并不太困难的工作，所以测试员往往容易上手，并且容易形成简单测试的概念。这里要介绍无障碍测试的另外一个概念——序列及交互化无障碍测试。

在功能测试中，同一个功能可能在不同的执行序列或在与不同的模块交互的过程中发生一些深层次的 Bug。虽然无障碍测试的编码对象主要是一些静态设置的属性，如快

捷键、Tab 顺序，HTML Tag 属性等，但在不同的开发、测试环境或浏览器下，这些静态属性却可能发生一些意想不到的变化，而这些变化往往不会在启动时出现错误，而往往是产品自身调用刷新等功能后出错。

所以，即使是如此直观的测试，还是需要充分考虑环境的特殊性，设计拓宽性思维的测试用例。

11.5 小　　结

本章一开始就明确地讲述了无障碍的诞生和发展历程，以及进行无障碍测试的原因。基于这些内容，本章以合理的顺序分别讲述了无障碍测试的基本概念、无障碍的相关标准、规范和无障碍测试的实践，还简单地介绍了当前主要的无障碍测试工具。

目前，国内网站的数量庞大，而且还在快速发展，但是又有多少网站符合无障碍的要求和按照无障碍标准在建设呢？可以说绝大部分的网站都还没达到无障碍的要求，无法让有障碍的人群和特定人群便利地浏览。无障碍测试的出现为网站建设和软件开发指明了方向。

本章内容新颖，作者试图通过无障碍的基本理念及一些技术、方法的介绍，来推动信息无障碍事业的发展。现在需要我们实实在在的努力，国内网站都需要根据无障碍的相关标准和规范来进行改进，今后的网站建设、软件的开发和测试，亟待按无障碍的相关标准和规范执行，以达到软件功能、网站及所有信息内容，对所有的人群都是无障碍的目的，更好地发挥所开发的软件和所有网站的功能，最大限度方便更多的人群获取信息。

习题与思考

1. 无障碍的软件和网站有利于哪些人群？
2. Web 无障碍中页面样式和字体应该注意哪些问题？
3. 软件无障碍的参考标准主要有哪些？
4. 简单介绍美国 508 条款。
5. WCAG 1.0 所倡导的理念是什么？
6. 要确保网页内容能良好的呈现，必须具备哪些关键要素？
7. 简要介绍 WCAG 2.0 以及它与 WCAG 1. 0 的不同之处。
8. Web 无障碍测试工具主要有哪些？
9. 简述无障碍软件的三个基本原则的具体内容。
10. 简述无障碍测试的一般步骤。

第6部分

案例分析

一个游客走进硅谷的一家宠物店，围着一个个宠物笼子看来看去。这时，有一个顾客走进店里，对营业员说："我想买一只C语言猴子。"营业员点点头，走到店角的一个笼子前，抓出一只猴子，找了一副项圈和皮带，交给那位顾客，说："五千美元一只。"那位顾客付了账，带着猴子走出了店门。游客惊讶地对营业员说："这么贵的猴子？普通猴子也就值两、三百美元，这一只怎么这么贵呀？""啊，这只猴子会编C程序，编得非常快，代码紧凑，没有Bug，它确实值这么多钱。"游客看着笼子里的另一只猴子说："这一只更贵了，要一万美元，这又是怎么回事？""噢，这是一只C++猴子，它可以用面向对象的方法编程，它会用Visual C++，还会一点Java，这可都是非常有用的技能呀！"游客四下转了转，突然瞧见了一只独占一个笼子的猴子。猴子的脖子上挂着价签，上面标着五万美元。游客这下惊讶得合不拢嘴了："这一只猴子比店里所有其他宠物加起来还要贵呢，它又能做什么？""唔，我也不知道它能做什么，但据说它是做咨询的。"

——《程序员》2008年第8期

这个寓言故事的题目叫《会编程序的猴子》。很多程序员、测试工程师积累了丰富的经验，就开始转行当顾问，从事咨询、培训等业务。最常用的就是运用案例分析给别人讲课，这样的咨询工程师工资是非常高的，因为他们几十年的经验是花大钱也买不来的。

本部分主要介绍基于RUP的软件测试案例，读者通过阅读案例，要在实践中学会应用。希望读者在总结所学知识和技能的基础上，敢于尝试创新，不但要理解案例分析，还要按照案例分析一步一步地开展软件测试实践，以提高自己的实际动手能力，使自己逐步成长为技术高超的测试工程师、测试技术专家。测试员的修炼之道，仍然是"从小工到专家"。

第12章 测试案例

本章主要讲解实际案例，要求读者在梳理、总结课程体系中的各个知识点和技能的基础上，针对不同的开发阶段，制定相应的测试计划，设计典型测试用例，使用软件测试技术和测试工具，达到测试目标，并进行回归测试，以实现软件测试各单项专业与技能整合运用的目标。

本案例共分三部分，第一部分是根据项目情况编写脚本，第二部分是使用 IBM Rational Test Manager（以下简称 TM）和 Robot 测试工具来完成实际项目，第三部分是使用无障碍测试工具来完成无障碍实例测试。

12.1 编写脚本

12.1.1 项目情况介绍

香港永隆银行要升级其银行业务系统，对总行及其在港的 34 家分行更换 300 多台柜员终端机。这套新系统是由 IBM 公司全球服务部所提供的 IBM 软件、个人计算机及服务器所组成，是一个基于 Java 技术而建立的全面集成系统。永隆银行业务解决方案采用 IBM 公司的 WebSphere Application Server 和 WebSphere Business Components Composer 为技术核心，有效加快建立集成式、多渠道零售银行业务应用模式。

此系统是以客户为中心的银行全面解决方案，将银行零售、资金、贸易融资、清算、电子商务和综合管理等信息系统全部综合为一体，形成一个完整的现代商业银行核心业务处理系统，同时通过有效的集成和整合、科学合理的迁移服务来确保银行业务经营的连续性及 IT 投资回报，并以此提升内部工作效率和改进客户服务，同时节省 IT 运营成本。

12.1.2 被测软件的特点

此银行业务系统软件规模庞大，子系统众多，外系统接口多，且业务功能还在不断扩展，软件本身不断升级，可以说该软件的测试任务繁重，回归性功能测试多，而且测试工作都是手工方式进行。

该业务系统采用 C/S 处理结构，包括前台系统和后台服务系统。其中，前台系统作为 Client 层节点，是业务数据的入口和最终出口。前台系统的业务功能明确，主要包括各类资金业务、信息业务的录入、发送、接收，数据查询以及用户管理等。

该软件的前台业务系统分为客户端和服务进程/程序两部分。其中，客户端软件基于 Windows 平台，提供了图形化操作界面。

12.1.3 测试入口的选择

作为数据输入和结果展示的窗口，被测软件的前台客户端是系统中较为稳定的部分。在软件升级过程中，程序的修改大部分集中于前台系统的服务进程/程序和后台服务系统，对客户端软件的修改量相对较少。同时，前台服务进程/程序、后台服务系统的具体功能又可以通过使用客户端的功能，构造不同的业务数据进行测试。因此，从该部分入手自动化测试，将降低自动化测试的开销，又使自动化测试的部件（测试脚本、数据、设置等）具有比较广泛的适用性，该部分是一个比较理想的自动化测试入口。

12.1.4 脚本编写

在开始自动化测试工作之前，像准备任何一次手工测试一样，要事先完成测试计划、测试设计等工作，文档化的测试用例准备就绪。下面重点描述自动化测试脚本实现过程。

（1）选择适合的测试用例和测试活动。针对被测软件特点，一笔资金业务的录入需要完成 20 多个输入操作，而且是多个测试用例的中间步骤，故选取该操作及相关用例进行自动化。

（2）准备输入数据文件。内容是将测试用例中的关键数据项抽取出来，生成数据文件。自动化测试工具大都支持 Excel 格式。按照测试需要，每行数据描述一个用例的测试业务及预期结果。笔者使用的一个数据文件如图 12-1 所示。

日期	业务种类	优先级	接收行	开户行	接受清算行	金额	返回码	提示
20040312	10：现金汇款	0：一般	001100001509	001100001509	001100001509	10	chlo0000	正确用例1-现金汇款
20040312	11：普通汇兑	1：紧急	001100001509	001100001509	001100001509	10	chlo0000	正确用例2-紧急汇兑
20040312	11：普通汇兑	0：一般	001100001509	001100001509	001100001509	12	chlo0000	正确用例3-普通汇兑

图 12-1　Excel 数据表

（3）录制测试过程生成自动化脚本工作。操作被测试的软件，执行一个正常的测试过程，录制出初始的自动化脚本。该脚本记录了所有的测试操作，包括鼠标移动、停顿和数据输入，成为实际需要的自动化脚本的初始框架。

由于一笔资金业务输入需要两个操作用户分别登录。分别录入、复核业务，本例分成两个脚本录制。

初始自动化脚本提供了开发自动化脚本的基础，但本身只能执行重复录制的操作。

（4）修改录制的自动化测试脚本。这是自动化测试的关键一步，其工作目标是得到满足测试需要的自动化脚本。该阶段的工作内容如下：

- 引用数据文件。在初始脚本的开始位置加入数据文件的引用。

- 引入循环控制。找出输入测试数据的段落，加入循环控制语句。
- 引用测试变量。选定需要从数据文件中取值的项引入测试变量。在循环体的开始位置对测试变量从数据文件的指定字段取值。在原来需要填入数值的位置换为测试变量。如资金业务数据录入脚本中，一笔业务的汇出行行号、日期、金额、汇款人账号、业务种类等，原来需要从界面手工录入，现在都可通过测试变量从数据文件读入。
- 引入检查点。根据被测软件的设计，服务进程在单击“确认”按钮后，会给出录入员/复核员录入数据的处理结果。分别在这两处设置检查点，检查服务进程处理是否正确。考虑被测试软件将处理结果会显示在弹出的信息框中，内容为返回码加解释信息，且返回码唯一，故选取预期返回码为对比关键字，进行结果检查。

自动化测试脚本调试，初步执行脚本，运行几组数据，查看脚本是否按预期的流程执行，执行过程是否合理。笔者在调试中遇到了中文字符不能输入、多个选项卡间无法切换和显示、鼠标操作的录制失败、暂停时间设置不合理导致回放失败等问题。由于涉及软件的使用配置，需要根据工具软件的使用说明进行调整，编制好的自动化测试脚本的主要内容不在这里赘述。

12.1.5 执行自动化测试

执行自动化测试脚本，完成自动化测试，首先应当注意合理协调和调度测试脚本间的关系。本例中由于被测软件在业务上要求复核员以交易序号为关键字进行复核录入，而该序号由录入员录入时系统自动给出，因此，必须先执行录入员脚本，人工获得交易序号后，记入数据文件，才能启动复核员脚本。另外，由于前台服务系统给出的交易序号都是顺序产生的，自动化测试脚本执行期间，不允许执行其他操作，否则将产生交易序号冲突，交易复核无法成功。

12.2 使用 TM 和 Robot

这里练习使用 TM 和 Robot 工具，被测程序 PetStore 是 Sun 公司在 Java2 平台上开发的一个应用程序样例，是 Java 软件在 J2EE 的蓝图程序，它示范了如何利用 J2EE 1.4 平台的性能去开发灵活、可升级的交叉平台企业应用系统。

PetStore 是一个运行在 Web 上的，为网络客户提供宠物信息浏览、网上订单和管理等功能的网上宠物店，在技术上使用了 J2EE 中的大部分企业组件和优秀的设计模式，提供了一套有高灵活性、扩展性和可升级的完善 J2EE 开发框架。

PetStore 系统利用 MVC 模式设计整个系统架构，将各层的对象清楚地分开。

PetStore 的目的是展示一个具有扩充性的企业运算架构，就是采用三层式设计：表示层在最外面，中间是执行企业运算逻辑的中间层组件，后端是存放数据资料的数据层。

Petstore 共有以下 4 个子系统组成。

① PetStore Storefront：是 PetStore 的核心，客户登录系统进行选择、定购、提交订单等。

② PetStore Admin：是系统的管理功能，包括销售统计、手工接受/拒绝订单。

③ PetStore Order Processing Center(OPC)：是订单处理中心，对客户提交的订单进行处理。

④ PetStore Supplier：是产品供应者提供维护功能。

本次实验的配置环境如下：

① J2EE PetStore 应用。

② SUN J2EE 应用服务器(版本 8.2)。

③ 数据库 J2EESDK 1.4 自带的 Derby 数据库。

④ Sun J2SDK 1.4.2。

⑤ 操作系统 Microsoft Windows XP Professional SP2。

⑥ CPU 类型 AMD Athlon 64，1600MHz(8x200)2600＋。

⑦ 系统内存 512MB(PC3200 DDR SDRAM)。

⑧ Internet Explorer 6.0.2900.2180(IE 6.0 SP2)。

⑨ 实验工具 Rational Robot、TM。

12.2.1 制定测试计划

RUP 是一个综合而详尽的流程框架，同样基于 RUP 的测试流程框架也是十分复杂的。本次实验主要是虚拟 RUP 构建阶段的部分测试工作流程。

首先，进行制定测试计划的活动。测试计划主要根据 RUP 测试计划模板生成，制定测试计划是测试管理工具 TM 的一个主要功能。测试计划的主要内容包括以下几个方面。

① 简介：对测试的目的、背景、范围以及项目核实进行介绍。

② 测试需求：是确定被测试对象的各项需求(如用例、功能性需求和非功能性需求)。此外，每阶段的测试工作要求，或者说所涉及的内容有可能不同。这种变化都将影响到测试需求的确定。通常参考的资料有软件需求规约和用例，也可能是一份包含用例的软件需求规约。

③ 测试策略：是确定对测试对象采用何种方法进行测试。上面“测试需求”中确定的是测试对象，而测试策略则说的是方法，即如何对测试对象进行测试。

对于每种测试，都应提供测试说明，并解释其实施和执行的原因。如果将不实施和执行某种测试，则应该用一句话加以说明，并陈述这样做的理由。例如，“此项不实施和执行某种测试。该测试不合适”。

制定测试策略时所考虑的主要事项有：将要使用的技术方法，预计测试何时完成以及其标准和使用的工具。

④ 资源：要列出推荐在测试 PetSoter 系统时使用的资源，包括系统资源和角色，主要职责、知识或技能。对于测试项目的系统资源，建议模拟生产环境，并在适当的情况下

减少访问量和数据库大小。

⑤ 项目里程碑：对 PetStore 的测试应包括前面确定的每一个测试工作所需的测试活动。应该为这些测试确定单独的项目里程碑，以通知项目的状态和成果。

⑥ 可交付工件(制品)：列出将要创建的各种文档、工具和报告，及其创建人员、交付对象和交付时间，包括测试模型、测试记录和缺陷报告。

⑦ 附录：项目任务：列出一些与测试有关的任务。

下面列出本次实验的测试计划中的一些重要部分。

1. 部分测试输入

测试计划编制的第一步是确定测试输入。测试输入是测试的依据或测试需要的验证。测试输入可以帮助决定需要测试的内容，还可帮助确定基于开发过程中可能需要的变化。迭代的开发过程是重要的，而在开发过程中的变化是一个频繁的必要的过程。

TM 有两个内置的测试输入类型：

① 在一个 Rational RequisitePro 项目中的需求。

② 在一个 Rational Rose 虚拟模型中的元素。

这些内置测试输入类型很容易获得需求和模型元素，并将这些输入与其他用以跟踪目的的测试资产相关联。

TM 支持常规的测试输入类型。例如，使用 Microsoft Excel 表格中的数值作为一个测试输入。在构建了测试列表，也就是确定了需要的测试之后，就可以创建测试用例了。测试用例限定了以测试输入为基础将要进行的测试。于是，需要把测试输入和测试用例结合起来对测试目的进行跟踪，在搭建起这些联系时，能够更容易地跟踪到这些测试输入的改变，而这些改变可能引起测试用例的变化或改变这些测试用例的实施。在本次实验中的测试输入中，只提供了 RequisitePro 中的一些软件需求输入，并与相关的测试用例联系起来，如图 12-2 所示。

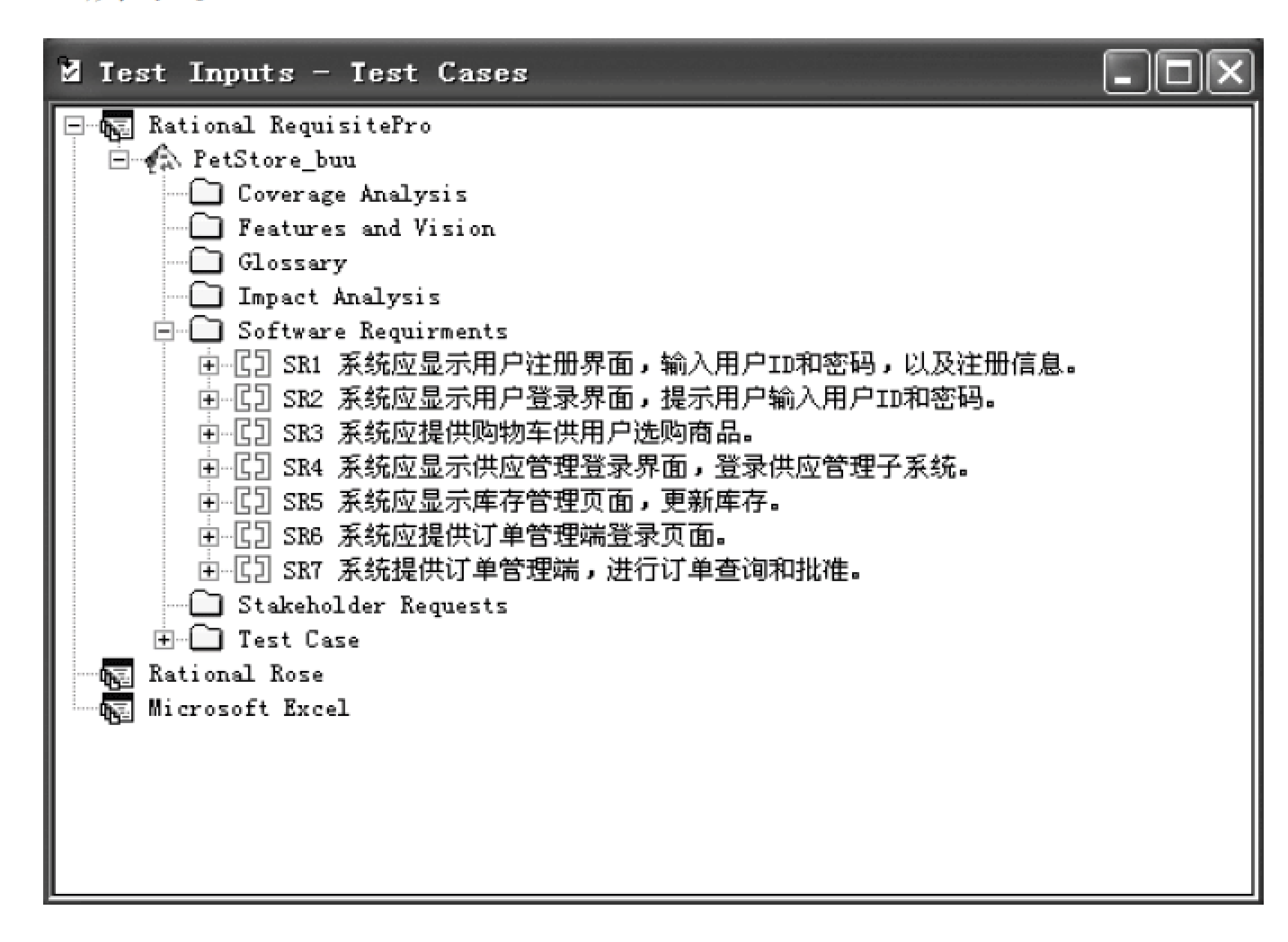


图 12-2　TM 中的测试输入

2. 部分测试需求

(1) 系统测试(功能测试)

对于功能测试需求,顾名思义,就是对于系统中描述的功能性行为有哪些地方需要测试。这部分内容主要是在项目早期通过对软件需求规约和用例的分析来获得的。对于PetStore系统的功能测试主要有:

- 核实Storefront子系统功能是否正常。
- 核实管理客户端子系统功能是否正常。
- 核实供应管理客户端子系统功能是否正常。

(2) 性能测试

性能测试需求来自于测试对象的指定性能行为。性能通常被描述为对响应时间和资源使用率的某种评测。对于本次实验中PetStore系统的性能测试主要有:

- Storefront子系统性能测试。
- 核实用户登录的响应时间。
- 管理客户端子系统性能测试。
- 核实管理客户端登录的响应时间。
- 供应管理客户端子系统性能测试。
- 核实供应管理客户端登录的响应时间。

(3) 负载测试

负载测试是属于性能测试之一,鉴于服务器性能的原因,只能做如下测试:

- 核实10个用户同时登录的系统响应。
- 核实50个用户同时登录的系统响应。
- 核实100个用户同时登录的系统响应。
- 核实10个用户同时提交订单的系统响应。
- 核实30个用户同时提交订单的系统响应。
- 核实50个用户同时提交订单的系统响应。

在一个测试计划里,可以通过创建测试用例文件夹来分层次组织测试用例。对于测试用例文件的分类准则,有以下几种方式:

- 对于项目中的每一个测试人员。
- 对于测试每一个种类或类型(单元测试、功能测试、性能测试和其他)。
- 对于系统的每一个用例。
- 对于应有程序的每一个主要模块。
- 对于测试过程的每一个阶段。

本次实验针对测试类型进行分类,针对以上的测试需求作了测试计划,如图12-3和图12-4所示。



图 12-3　TM 中的功能测试计划



图 12-4　TestManager 中的性能测试计划

3. 部分测试策略

(1) 系统测试(功能测试)

应用程序测试应该集中在可以被直接追踪到用例(或业务功能)和业务规则的目标需求。这些测试的目标在于核实能否正确地接收、处理和检索数据,以及业务规则是否正确实施。这种类型的测试基于黑盒技术,即通过 GUI 与应用程序交互并分析输出(结果),来验证应用程序(及其外部进程)。对于该系统的测试方法如表 12-1 所示。

表 12-1 系统测试策略

项	说　　明
测试目标	确保系统导航、数据输入、处理和检索正确
技术	利用有效的和无效的数据来执行各个用例、用例流或功能，以核实以下内容： • 在使用有效数据时得到预期结果； • 在使用无效数据时显示相应的错误消息或警告消息； • 业务规则都得到了正确应用
完成标准	所有的计划测试已全部执行，所有确定的缺陷已得到处理
需考虑的特殊事项	无

（2）性能测试

性能测试评测响应时间、事务处理速率和其他与时间相关的需求。性能测试的目的在于核实和确认性能需求是否已经得到满足。执行最初的测试时应该使用“额定”负载，该负载与目标系统所使用过（或预期的）的正常负载相近。第二次性能测试则使用最大负载。对于该系统的性能测试方法如表 12-2 所示。

表 12-2 性能测试策略

项	说　　明
测试目标	核实系统所指定的事务或业务功能在以下情况下的性能行为： • 正常的预期工作量； • 预期的最繁重工作量
技术	• 使用为功能或业务周期测试制定的测试过程； • 通过修改数据文件来增加事务数量，或通过修改脚本来增加每项事务的迭代数量； • 脚本应该在一台计算机上运行（最好是以单个用户、单个事务为基准），并在多个客户机（虚拟的或实际的客户机）上重复
完成标准	• 单个事务或单个用户在每个事务所预期或要求的时间范围内，成功地完成测试脚本，没有发生任何故障； • 多个事务或多个用户在可接受的时间范围内能成功地完成测试脚本，没有发生任何故障
需考虑的特殊事项	通过创建“虚拟的”用户负载来模拟许多个（通常为数百个）客户机。此负载可通过“远程终端仿真”（Remote Terminal Emulation）工具来实现

（3）负载测试

负载测试是一种性能测试。在这种测试中，将使测试对象承担不同的工作量，以评测和评估测试对象在不同工作量条件下的性能行为，以及持续正常运行的能力。负载测试的目标是确定并确保系统在超出最大预期工作量的情况下仍能正常运行。此外，负载测试还要评估性能特征。对于该系统的负载测试方法如表 12-3 所示。

表 12-3　负载测试策略

项	说　　明
测试目标	核实所指定的事务或商业理由在不同的工作量条件下的系统响应时间
技术	通过修改数据文件来增加事务数量，或通过修改测试来增加每项事务发生的次数
完成标准	多个事务或多个用户在可接受的时间范围内成功地完成测试，没有发生任何故障
需考虑的特殊事项	负载测试应该在专用的计算机上或在专用的机时内执行，以便实现完全的控制和精确的评测

12.2.2　测试设计与实施

1. 部分功能测试设计与实施

(1) 用户订购功能测试设计

用于功能测试的测试用例来源于测试目标的用例，应该为每个用例场景编制测试用例。用例场景要通过描述流经用例的路径来确定，这个流经过程要从用例开始到结束遍历其中所有基本流和备选流。为了从用户订购用例生成测试用例，下面从 PetStore 用户子系统的购买用例的事件流入手进行介绍。

① 基本流-登录。

步骤 1，当主角在浏览页面中选择好了准备购买的宠物，单击 Check Out 链接，本用例就开始了。

步骤 2，系统需要用户提供识别信息，用户提供以下信息：

§ User Name

§ Password

系统验证用户信息有效。

步骤 3，系统显示用户信息，用户确认后单击 Submit 按钮。

步骤 4，用户完成订购，系统显示订单 ID。系统发送订单信息到订购处理中心。本用例结束。

② 备选流。

- 无效的用户 ID/密码。如果在基本流步骤 2 中，系统无法找到用户 ID 或密码无效，此时就会显示一个错误信息。用户订购信息被清除，用例结束。
- 用户已经登录。如果在基本流步骤 2 中的用户已经登录，系统会跳过基本流步骤 2，用例继续基本流步骤 3。
- 用户关闭浏览器。用户在基本流步骤 3 之前任何时候关闭浏览器，用户订购信息被清除，此时本用例结束。

③ 前置条件。

- 用户必须至少选择一种宠物购买。

• 用户必须拥有合法的用户 ID 和密码。

可以从这个用例生成下列场景，如表 12-4 所示。

表 12-4　用户订购用例场景

场　景　1	成 功 订 购	基　本　流	
场景 2	用户 ID/密码无效	基本流	备选流 1
场景 3	用户已登录	基本流	备选流 2
场景 4	用户关闭浏览器	基本流	备选流 3

对于以上的场景，每个场景至少要确定一个测试用例，如表 12-5 所示。

表 12-5　用户订购测试用例

测试用例名称	场景/条件	用户 ID	密码	预 期 结 果
用户订购 1	场景 1-成功订购	user	1234	用户完成订购系统显示订单 ID
用户订购 2	场景 2-用户 ID/密码无效	user	1243	系统无法找到用户 ID 或者密码无效，就会显示一个错误信息
用户订购 3	场景 3-用户已登录	user	1234	系统显示用户信息，用户确认后单击 Submit 按钮
用户订购 4	场景 4-用户关闭浏览器	/	/	/

（2）用户订购功能测试实施

本次实验中功能测试实施主要是使用 TM 和 Rational Robot 工具，针对得到的测试用例，生成相应的 GUI 测试脚本。主要步骤介绍如下。

① 录制测试脚本。

根据已获得的测试用例，使用 Robot 录制相应的测试脚本。需要注意的是在每次录制之前应把测试环境设置好（例如删除 IE 的 Cookies 等）。以下是其中测试脚本 RS_用户订购 1 的代码：

```
Sub Main
Dim Result As Integer
'Initially Recorded:2006-3-7 12:32:37
'Script Name: RS 用户订购 1
StartBrowser"http://localhost:8080/petstore","Window/Tag=WEBBorwser"

Window SetContext,"WindowTag=WEBBorwser",""
Browser NewPage,"HTMLTitle=Java[tm]Pet Store Demo Petstore", ""
HTMLLink Click,"HTMLText=enter the store",""
Browser NewPage,"HTMLTitle=Welcome to the BluePrints Petstore",""
HTMLLinkn Click,"HTMLText=Dogs",""
Browser NewPage,"HTMLTitle=Items",""
HTMLLink Click,"HTMLText=Bulldog",""
Browser NewPage,"HTMLTitle=Product",""
HTMLLink Click,"HTMLText=Add to Cart;Index=2",""'把选择的宠物加入购物车
```

```
Browser NewPage,"HTMLTitle=Cart",""
HTMLLink Click,"HTMLText=Check Out",""'对已选择的宠物进行购买
Browser NewPage,"HTMLTitle=Sign On",""
EditBox Left_ Drag,"Name=j_username","Coords=60,13,-41,26"
InputKeys"user"'输入用户 ID
EditBox Left_Drag,"Name=j_password","Coords=55,11,-70,19"
InputKeys"1234"'输入密码
PushButton Click,"Name=submit"

Browser NewPage,"HTMLTitle=Enter Order Information",""
Result=PushButtonVP (CompareProperties,"HTMLText=Submit","VP=Object
Properties")'查证点 1:验证用户登录成功
PushButton Click,"HTMLText=Submit"
Browser NewPage,"HTMLTitle=Order Placed",""
Result=HTMLLinkVP (CompareProperties,"HTMLText=Sign out"VP=Object
Properties2")'查证点 2:验证提交订单成功
Window CloseWin,"",""
End Sub
```

② 编辑测试脚本。

必要时应对已录制的测试脚本进行编辑和扩展,以及插入验证点。本次试验由于录制功能简单,在录制过程中就已经在测试脚本中安排插入了验证点。

由测试用例生成相应的测试脚本以及验证点如表 12-6 所示。

表 12-6　登录功能测试脚本

测试用例名称	测试脚本名称	验　证　点
用户订购 1	RS_用户订购 1	验证用户登录成功;验证提交订单成功
用户订购 2	RS_用户订购 2	验证用户密码无效
用户订购 3	RS_用户订购 3	验证用户登录成功;验证显示用户信息;验证提交订单成功
用户订购 4	RS_用户订购 4	验证显示用户信息

另外对于验证点,特别是对于采用对象捕捉方式获得的验证点,应该编辑其捕捉到的众多属性,去除容易引起测试失效的属性(如弹出的对话框的位置、大小等),保留最关键的属性(如弹出的对话框的名称等),这样做是为了能增加验证点的健壮性。

③ 创建测试组(suites)。

测试组是另一个在 TM 中实施测试的方法。一个测试组展示了要测试的工作的一个分层描述,或要添加到系统中的工作量。它展示了诸如用户或计算机组之类的条目,每个组的资源分配,执行哪个组的测试脚本,以及每个测试脚本执行的次数。

可以在功能测试中使用测试组,也可以在性能测试中使用它。虽然在两种测试类型中,测试组的概念是相同的,但是将要插入不同的测试组条目,并选择不同的选项,则依赖于是否正在执行一个功能测试或性能测试。

对于用户订购功能,最后确定一个测试用例"用户订购功能",并在 TM 中用一个包

含上面4个测试脚本的测试组"用户订购功能组"实现，如图12-5所示。



图12-5　用户订购测试用例和测试组

本次实验的功能测试用例全部采用测试组方式实施，这样做有利于对实施测试用例的测试脚本进行修改和维护。

2. 部分性能测试设计与实施

(1) 用户登录性能测试设计

性能测试用例的主要输入是补充规约，其中包含非功能性需求。对于补充规约内阐明性能标准的各条说明，都至少要确定一个测试用例。性能标准通常表示为时间/事务、事务量/用户或百分数的形式。

对于部分性能测试用例的设计如表12-7所示。

表12-7　性能测试用例

测试用例名称	负　载　量	条　　件	预 期 结 果
10用户同时登录	10个用户	成功登录	系统响应时间在1秒内
50用户同时登录	50个用户	成功登录	系统响应时间在5秒内
100用户同时登录	100用户	成功登录	80%以上用户成功登录，系统响应时间在10秒内
10用户同时提交订单	10个用户	成功提交	系统响应时间在5秒内
30用户同时提交订单	30个用户	成功提交	系统响应时间在10秒内
50用户同时提交订单	50个用户	成功提交	80%以上用户成功提交

(2) 负载测试实施

本次实验中负载测试实施，主要是使用TM和Rational Robot工具，针对得到的测试用例，生成相应的VU测试脚本。主要步骤介绍如下。

① 录制 VU 测试脚本。

在录制用于性能测试的 VU 脚本时,也同样要注意测试环境的准备。在 VU 脚本录制过程中,要根据测试的需要插入同步点(Synchronization)、计时点(Timer)、阻塞点(Block)。同步点是为了控制让虚拟测试者同时完成一个动作,例如同时登录、同时提交订单的同步点。计时点是为了模拟真实的客户端与服务端的交互时间,它同时考虑了用户的思考时间以及系统的响应时间。阻塞点则与计时点基本类似,只是它不考虑用户的思考时间,只计算事务处理的时间。例如,要测试多用户同时登录的性能,先录制一个名为 userlogin_perf 的 VU 脚本。在录制过程中,在输入用户 ID 和密码后,插入一个同步点 login_sync,之后再插入一个阻塞点 login_timer。然后,在按下按钮 Sign In 后,出现登录成功页面时,停止阻塞点 login_timer。这样在不同数量的虚拟者执行脚本时,就可以统计出登录的响应时间。

② 设置和应用数据池。

对于负载测试要求模拟不同的用户同时完成测试脚本,这就需要使用数据池(Datapool)功能。数据池在 Rational 测试中使用率很高,同时也充分体现了自动化测试的优势。使用数据池,可以通过简单的脚本完成大量数据的测试,缩短测试时间、提高测试效率和测试质量。

TM 具有创建和管理数据池的功能,在创建数据池时应对字段名称、数据类型等属性进行设置,然后再生成数据。完成了数据池的创建,还可以执行数据池的编辑、改名、删除、导入、导出等操作。数据池有数据生成能力,但是不具备数据的计算能力。这种情况下可以利用其他工具(如 Excel)生成数据,并保存为 csv 格式,然后在 Manage Datapools 中导入。

本次实验创建了一个拥有 100 个用户信息的数据池用于测试。

在 GUI 脚本中使用数据池需要手工编码,Robot 不能自动生成。主要步骤就是打开数据池,从数据池中读取相应记录赋予相应的变量,最后要关闭数据池。

在 VU 脚本中的数据池可以通过录制自动产生,如果不能自动产生可以手工编码加入。当然也可以把自动产生的数据池用按需要创建的数据池替代。这就需要通过 Datapool information 进行配置,如图 12-6 所示。

③ 创建测试组。

在性能测试中,一个测试组不仅能够执行测试脚本,也可以模拟用户的活动,添加工作量到一台服务器中去。一个测试组可以如同一个虚拟者执行一个测试脚本那样简单,也可如同上百个虚拟者在不同的组内,每组在不同的时间内执行不同的测试脚本。例如,在本实验中,测试 10 个用户同时登录的系统响应时间,就创建一个测试组"性能组_10 用户同时登录",并在该组中添加一个数量为 10 的用户组,在该用户组下插入先前录制好的脚本 userlogin perf。

本次实验的性能测试用例的实施同样采用测试组方式。

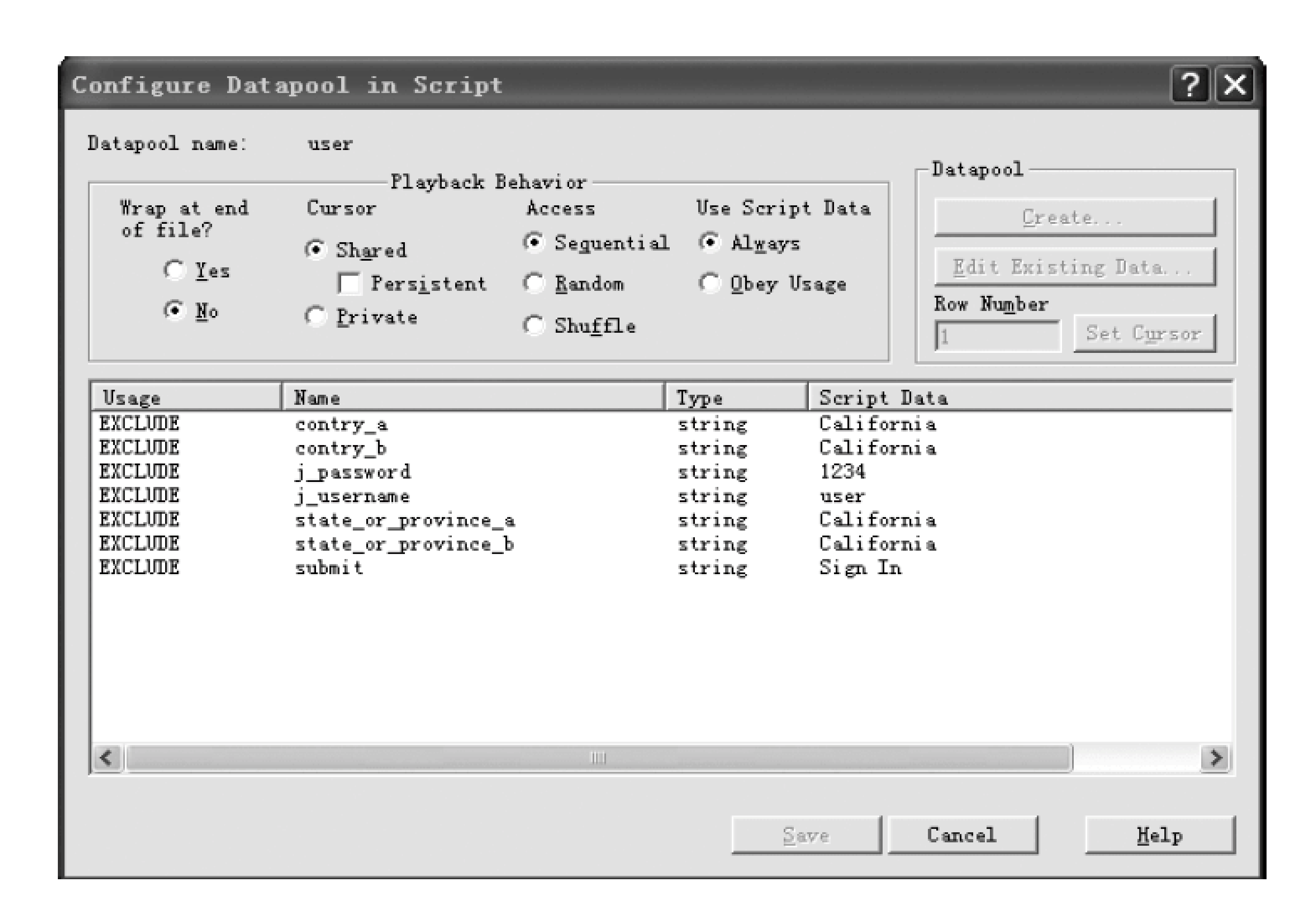


图 12-6 数据池设置

12.2.3 测试执行

在 TM 中执行测试,主要通过以下几种方式:

- 自动测试脚本;
- 手动测试脚本;
- 测试用例;
- 测试组。

这里主要讨论测试的自动执行,也就是自动测试脚本的执行。TM 工具能够根据测试的目的,以不同的方式执行以上生成的测试脚本。可以把测试脚本与相应的测试用例联系起来,然后通过执行测试用例来执行测试脚本,也可以把测试脚本与相应的测试组联系起来,然后通过执行测试组来执行测试脚本。

1. 功能测试执行

在本次实验中所有的功能测试执行都是通过测试用例方式进行,前提是待执行的测试用例都用相应的测试脚本和测试组,而且它们在此前已经实施过了。TM 可以通过对功能测试计划的执行来执行其包括的所有测试用例。所有的测试用例都与相应的迭代过程联系,执行测试计划时应选择以相应的迭代过程执行测试计划。本次实验的功能测试用例,全部与构建迭代过程联系在一起,所以执行功能测试计划时选择构建迭代。

2. 性能测试执行

在本次实验中所有的性能测试执行都是通过测试组方式进行,前提是待执行的测试组都组织了相应的测试脚本和测试用例以及相应的用户组。

12.2.4 测试评估

1. 测试日志

在执行完一个测试组、测试用例或测试脚本之后，TM 将把结果写入到一个测试日志中。使用 TM 的 TestLog 窗口查看在执行完了一个测试组、测试用例或测试脚本之后，创建测试日志。

功能测试执行完毕时，显示的测试日志如图 12-7 所示。

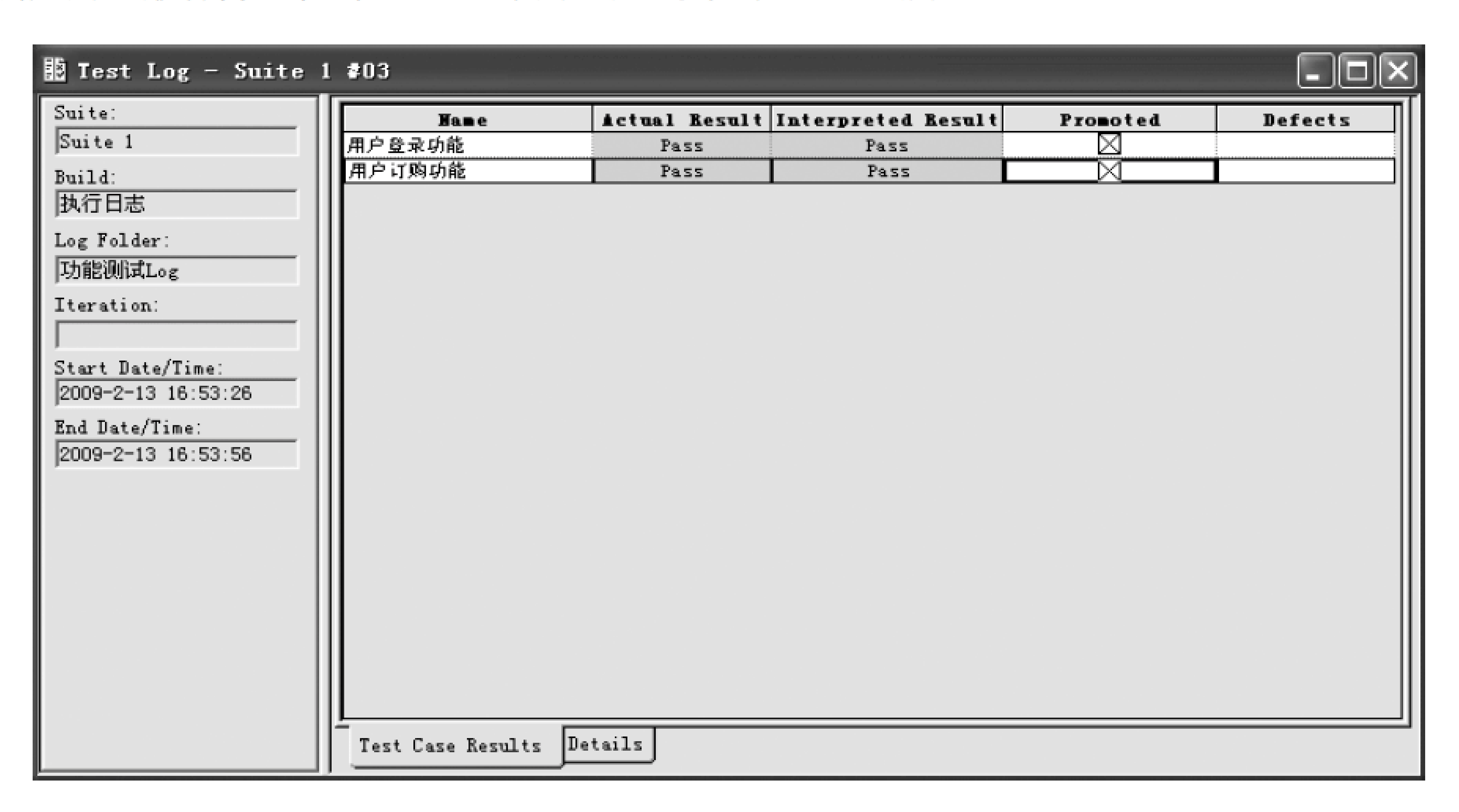


图 12-7　功能测试计划执行日志

从测试日志(Test Log)窗口中，可以单击测试用例结果(Test Case Results)标签来获得每个测试用例总的结果，是通过还是失败。测试用例结果(Test Case Results)标签展现一个测试用例的执行结果，或者包含了测试用例的一个测试组。然后提交通过的测试用例。性能测试组执行完毕，显示的测试日志如图 12-8 所示。

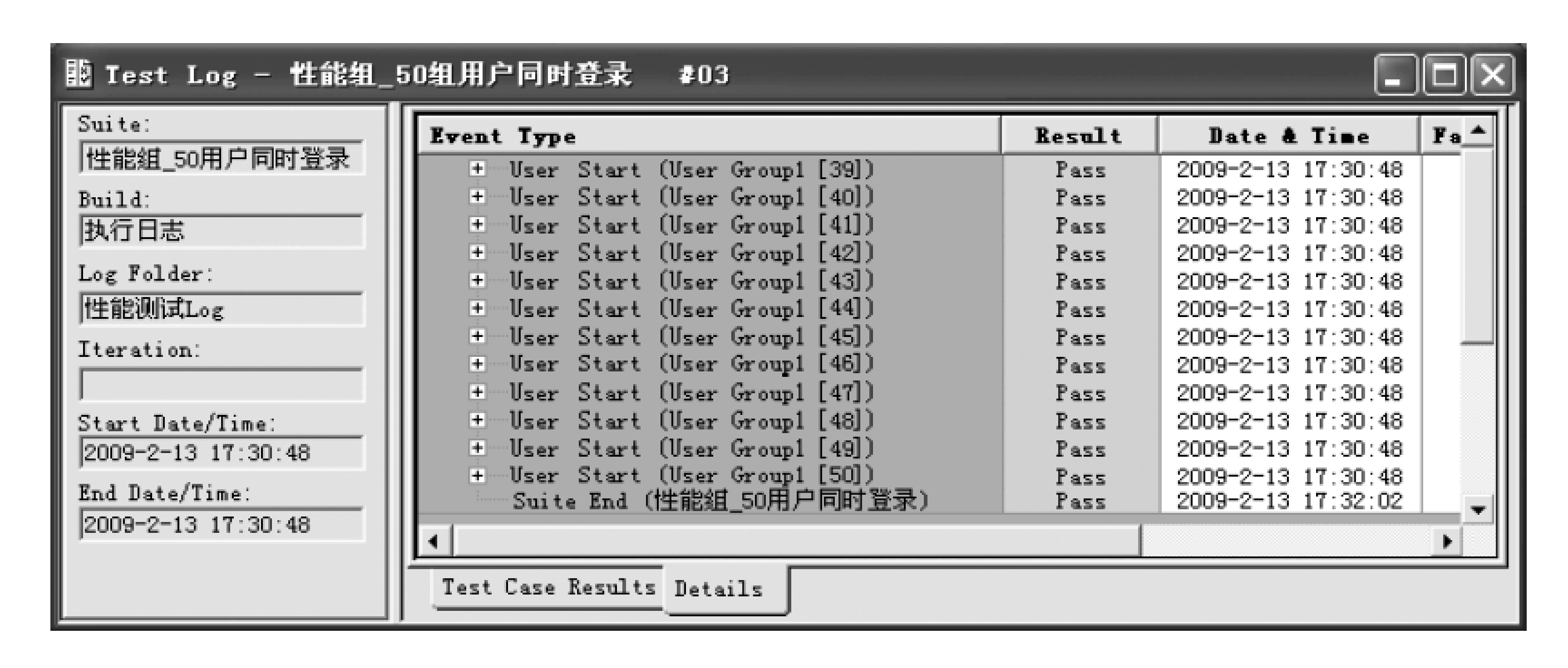


图 12-8　50 个用户同时登录性能测试日志

从图 12-9 得出 30 个用户同时提交订单时，系统的平均响应时间为 2.31 秒，标准偏

差为 4.84 秒。符合相应性能测试要求。

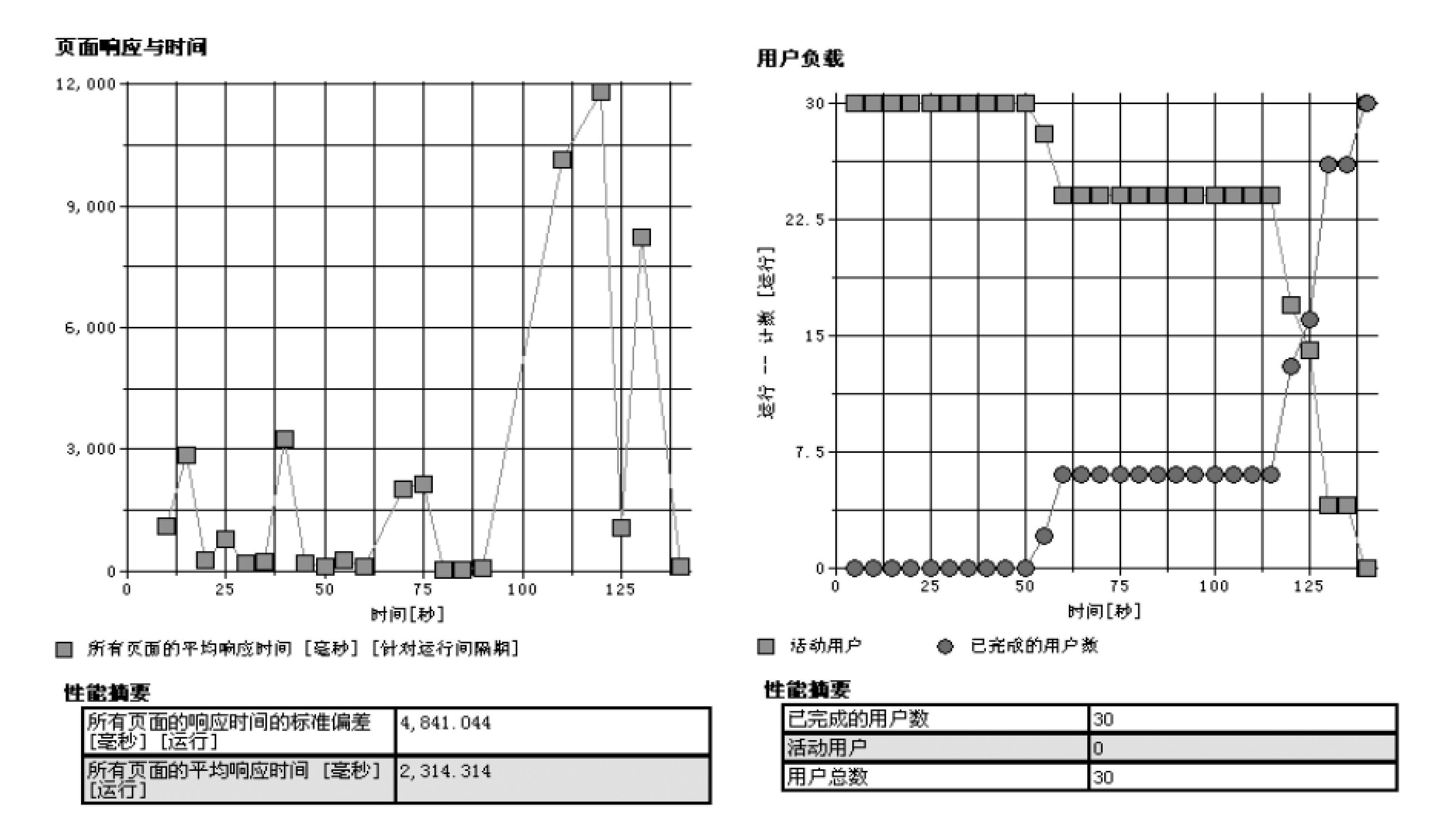


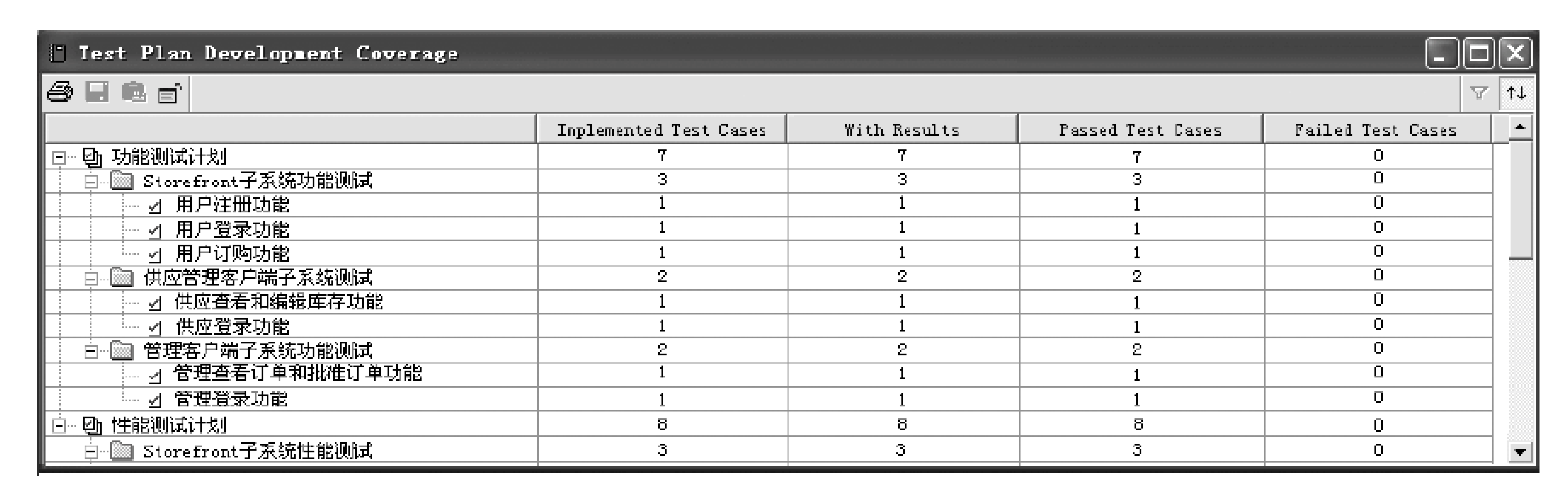
图 12-9　30 个用户同时提交订单性能测试日志

2. 测试报告

TM 中可生成的报告主要有测试用例报告(Test Case Report)、性能测试报告(Performance Testing Report)和列表报告(Listing Report)三类。

(1) 测试用例报告

测试用例报告帮助跟踪计划实施的过程，以及测试用例的执行。这些报告有多种展现格式，包括栏(bar)图、堆(stack)图、面积(area)图、线(line)图、饼(pie)图和树(tree)图。限于篇幅，本文只列出本实验的测试计划执行覆盖报告(Test Plan Execution Coverage Report)。测试计划执行覆盖报告，显示计划的测试用例的实施数目，有结果的测试用例数目、通过的测试用例数目，以及失败的测试用例数目，如图 12-10 所示。


Test Plan Development Coverage

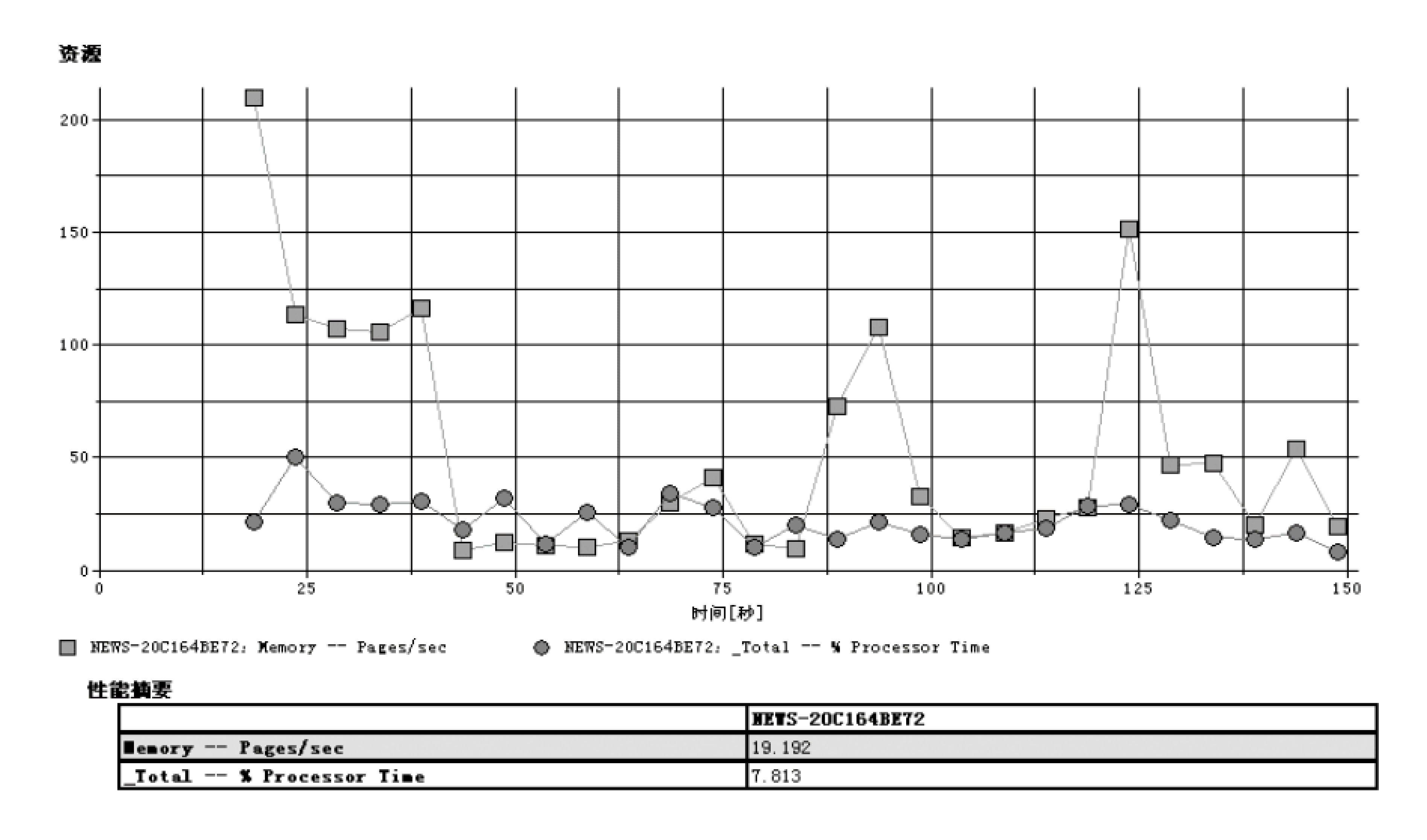
	Implemented Test Cases	With Results	Passed Test Cases	Failed Test Cases
功能测试计划	7	7	7	0
Storefront子系统功能测试	3	3	3	0
用户注册功能	1	1	1	0
用户登录功能	1	1	1	0
用户订购功能	1	1	1	0
供应管理客户端子系统测试	2	2	2	0
供应查看和编辑库存功能	1	1	1	0
供应登录功能	1	1	1	0
管理客户端子系统功能测试	2	2	2	0
管理查看订单和批准订单功能	1	1	1	0
管理登录功能	1	1	1	0
性能测试计划	8	8	8	0
Storefront子系统性能测试	3	3	3	0

图 12-10　测试计划执行覆盖报告

(2) 性能测试报告

性能测试报告能帮助分析在指定条件下一个服务器的性能。例如，可以统计一个虚拟测试者花费多长时间来执行一个命令，以及在不同的测试组执行的响应时间的变化。

性能测试报告也有几种类别，例如在测试组执行完毕时，显示的性能报告（Performance Report）和命令状态报告（Command Status Report），在上面的测试日志图表中就有这两类报告。图 12-11 显示了资源利用情况，如内存和 CPU 的使用情况。



	NEWS-20C164BE72
Memory -- Pages/sec	19.192
_Total -- % Processor Time	7.813

图 12-11　资源情况图

12.3　无障碍测试

12.3.1　项目背景

雅典奥运会由于信息无障碍建设不完善，会后收到了不少投诉。悉尼奥运会筹备工作周密完善，成功地举办了一次受到全世界普遍好评的盛会。中国互联网协会联合中国残疾人福利基金会、中国通信标准化协会共同发起“北京 2008 奥运会、残奥会网站信息无障碍行动”，时间从 2008 年 3 月持续到 2008 年 9 月底。无障碍行动的总体目标是以 2008 奥运为契机，在奥运会开幕前，通过政府部门强有力的推介，业内各方的积极参与，完成奥组委、中残联等有重大影响力的网站信息无障碍化改造，同时推动相关政府网站以及部分知名门户网站，凡与奥运主题相关内容初步达到信息无障碍的要求。

为了更好地满足残疾人和特定人群了解北京奥运会和残奥会的信息，中国残联网进行了全面的无障碍测试，有的放矢地进行无障碍改造和有针对性地改版。

12.3.2　测试流程

1. 进行培训

首先需要对测试团队进行信息无障碍培训。信息无障碍刚刚在国内发起，还尚未普

及，人们对信息无障碍的了解和认识还有些陌生。所以网站的开发人员，有必要对他们进行培训。

在北京 2008 奥运会、残奥会网站信息无障碍行动中，中国互联网协会专门召开了几次交流会。会上媒体和企业纷纷就网站信息无障碍案例和标准进行了探讨，工业和信息化部电信研究院吴英桦高级工程师就网站信息无障碍改造工作进行了详细的讲解、布置和要求，他还组织编写了《信息无障碍化网站标准实施指南》，对参与网站的技术人员进行了培训，并对网站改造的全过程给予了指导和技术支持。

2. 确定标准

第二次中国信息无障碍标准推进工作研讨会上介绍了中国通信标准化协会正在抓紧制定信息无障碍标准，但还没有正式出台。可用的标准有 WCAG 1.0、美国 508 条款等。

值得一提的是我国开展信息无障碍研究和宣传工作有五年多了，已经取得了可喜的进展，信息无障碍理念得到社会各界的认可，并开始达成共识，逐步得到推广和普及，信息无障碍建设方面的技术创新成果也不断涌现。有关部门正在制定中国信息无障碍技术标准体系，即将有我们国家的无障碍标准和规范，另外，正在修改的老年法也增加了老年人信息无障碍的内容。相信信息无障碍随着社会的文明进步将会得到进一步发展，一定会受惠于更多的人群。

3. 静态测试

首先目测网站，可以发现此网站要达到 WCAG 1.0 第一优先级还存在着很多问题。这里仅列出部分问题，以 WCAG 1.0 条款作说明。

WCAG 1.11：H101210 以无障碍多媒体来替代 ASCII 文字艺术。

WCAG 1.14：H101213 多媒体呈现时，必须提供语音说明。

WCAG 1.15：H101214 多媒体呈现时，必须同步产生相对应替代的语音或文字说明。

WCAG 4.1：H104200 明确地指出网页内容中语言的转换。

WCAG 7.1：H107200 确保网页设计不会引起屏幕快速闪烁。

WCAG 11.1：H111200 如果不能使这个网页无障碍化，需提供另一个相等的无障碍网页。

WCAG 14.1：H114200 网页内容要使用简单易懂的文字。

此外，还可以借鉴 IBM 软件无障碍检测表，对照检测表上的检查点逐一审核测试网站的各个页面和功能，以此发现网站的无障碍问题。

4. 工具测试

常用的工具有 IBM 公司的 Rational AppScan 工具，不管是哪个工具，都参照了 WCAG 1.0 标准，因此检测得到的结果都是一样的。图 12-12 是某测试工具对中国残联网站（旧版）进行测试后得到的结果。

Uh-oh! There are 508 problems!

The verdict is in and unfortunately the page you tested does not adhere to the WAI accessibility guidelines we evaluated it against.

There were 508 problems identified on the page you submitted. These issues are broken out by WAI priority in the table below.

Accessibility issues for ÖÐ¹ú²Ð¼²ÈËÁª°Ï»á by WAI Priority

Priority 1 - "must fix"	0 problems
Priority 2 - "should fix"	505 problems
Priority 3 - "may fix"	3 problems

- Priority 1 issues *must* be fixed to provide the most basic level of accessibility.
- Priority 2 issues *should* be fixed to provide the minimum level of accessibility recommended by the EU.
- Priority 3 issues *may* be fixed to maximise accessibility.

图 12-12　测试工具进行测试得到的结果

图 12-13 是某网站对中残联网站(旧版)进行测试得到的结果,可以看得出来测试结果很直观,明确地发现了问题所在。

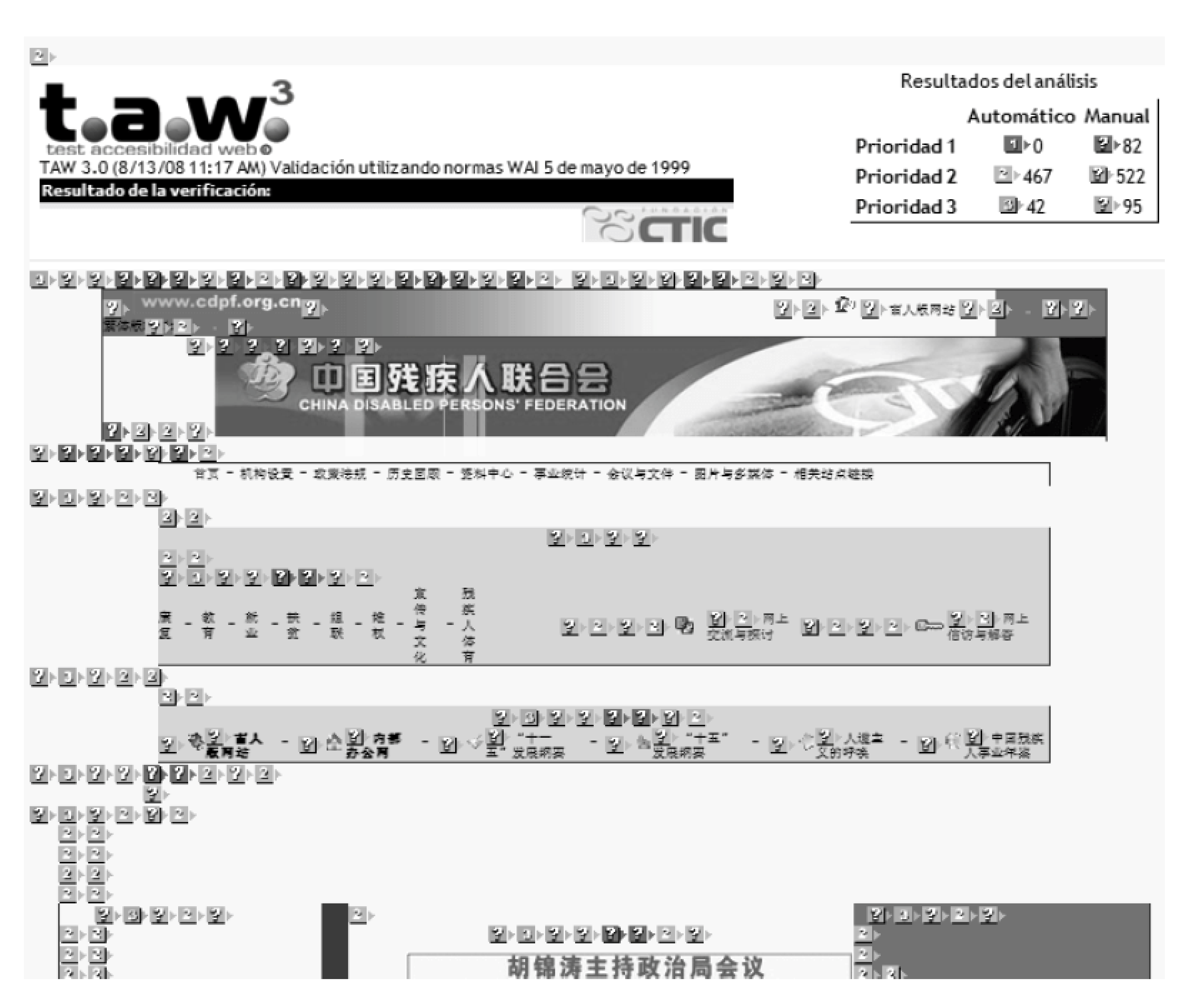


图 12-13　某网站对中残联网站(旧版)进行测试得到的结果

这里只列出测试工具所发现的部分问题:

① 使用了嵌套表格。因为语音浏览器或者屏幕阅读器阅读表格是按照顺序读的,一般是从左到右,自上而下,按顺序排列而有规律的表格符合这样的要求,而嵌套表格显然不行。

② 相邻链接被空格分开了。当相邻链接被空格分开时,一些辅助技术很容易把相邻

链接当作一个链接。

③ 发现了不推荐使用的<font>元素标记。W3C 组织出台的 HTML 最新标准已不再支持这个标记。这意味着浏览器和辅助技术不会再支持这个标记,因此可能会导致无法访问其内容。还有其他不推荐使用的元素标记也要注意。

5. 进行改造设计

将测试得到的结果反馈给开发人员,要求开发人员根据测试结果和建议进行改造设计。以上仅是初步完成了无障碍测试流程,按照 RUP 最佳实践经验,还要进行反复循环迭代。按照测试反馈得到的意见进行设计和改造,改造后的网站表面上应该通过了检查,至少要通过目测检查,没有明显的无障碍问题。更深层次的检查可利用工具检查来继续完成检测,以确认所有的问题都已改正,确保网站达到无障碍标准,即 WCAG 1.0 第一级。这个时候可能还会再发现问题,接着再反馈给开发人员,重新循环迭代。当然也可以把无障碍测试作为整体测试的一个环节来做,例如无障碍测试通常和功能测试一起做,也可以放在易用性测试里面进行测试。

6. 验收测试

在适当的时候邀请专家进行审查,或者请用户来进行使用测试,这一般是正式的无障碍测试,在这个测试里,专家将对网站进行验收审查,邀请的用户要在测试地点使用专门的辅助工具,例如使用读屏软件来测试网站,并由实验人员来现场观察和记录评估者的行为。当然这样的测试也可以放在无障碍测试流程里做,而不必放在最后。

以上流程只是针对现有网站进行无障碍改造而设计的,一般按照 RUP 原则,在软件或网站项目初期,就应该在需求分析以及设计阶段就将无障碍的需求考虑进去,并在测试中对其进行验证,一般是根据角色的需求在程序中添加检查点,通过 RUP 的自动化测试功能来对无障碍功能进行测试。

12.3.3 无障碍改造

以下是中国残联网站的声明。

方式一,全网站遵循 WCAG 2.0 进行无障碍网页设计,符合 XHTML 1.0 技术规则,适用于盲用读屏软件。

① 全网站页面可用键盘操作,不限于鼠标。

有些用户可能因为自身的某种缺陷而不能使用鼠标,有些可能因为自己的计算机没有配备鼠标等设备。只有网站实现了可以通过键盘操作来完成网站的上述的操作功能,这些用户才可以无障碍地访问该网站。此项改造是为了使这些用户便利地浏览网页。

实现方法如下:

- 在链接、表单控制及对象间,提供合乎逻辑的 Tab 条约顺序。例如在 HTML 里,可以通过 tabindex 属性来制定 Tab 顺序,确保页面设计的逻辑性。
- 为重要链接提供键盘快捷键(包括客户端图像映射、表单控制、表单控制群组)。

例如在 HTML 里，可以使用 accesskey 来指明键盘快捷键。

② 网站设置导盲砖快速键(:::)，使用者可快速到达各主要区块。

所谓导盲砖(即定位点)，其显示方式是利用三个冒号(:::)来代表，且需要搭配快速键设计来使用。主要用途在于帮助使用者快速定位及搜索，其有以下几个优点：

- 快速跳跃至网页不同区块，可避免使用者迷失在网页中。
- 方便使用者在各框架页(frame)间快速移动。
- 可引导使用者依自己需要，跳至所需的区块中。

以下是设计导盲砖的一个例子：

```
<a accesskey="Alt+L" href="intro.jsp" title="左侧区域">:::</a>
<a accesskey="Alt+C" href="intro.jsp" title="中间区域">:::</a>
<a accesskey="Alt+R" href="intro.jsp" title="右侧区域">:::</a>
```

Title 属性文字方便语音合成器告知使用者是在网页中的哪一个区块。

③ 网页图片均标示文字说明(即可替代性文本)。

有些用户可能无法看到图片；有些可能使用基于文本，不支持图片的浏览器；有些可能关闭了图片支持(例如由于低速网络连接问题)。因此，这些用户就无法获得这些图片所带来的信息。可替代性文本的好处在于它可以被绝大部分技术或工具，通过不同方式呈现给具有各种障碍的残疾人。文字可以被合成语音或被翻译为盲文，或在屏幕、纸上以不同尺寸显示。合成语音对于视觉障碍、阅读障碍、理解和学习障碍都非常重要。而盲文也利于盲人理解内容，而且文字显示对于聋人和绝大部分 Web 使用者都非常实用。因此，中国残联网为所有的图片添加了文字说明。

实现的方法：为 IMG 元素提供 alt 属性替代文字。对于较复杂的、alt 属性无法完整描述的替代内容，可利用 IMG 元素的 longdesc 属性，如图 12-14 所示。



图 12-14　添加文字说明

④ 所有链接均添加提示文字。

为链接添加文本与为图片添加文本的作用是类似的，实现的方法与为图片添加文本的实现方法一样，如图 12-15 所示。

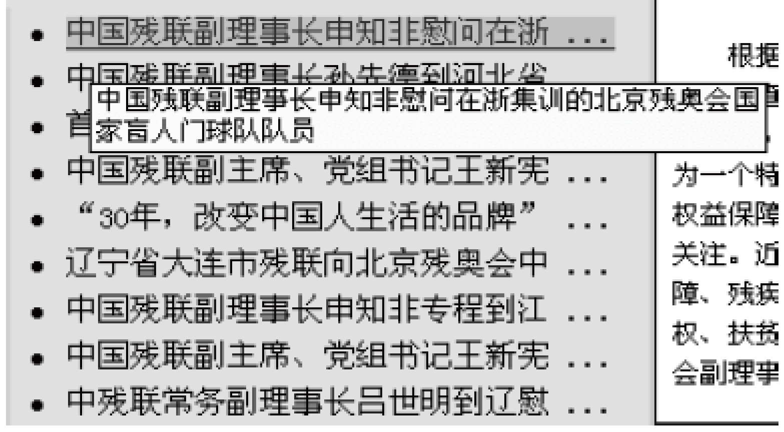


图 12-15　添加提示文字

⑤ 对于网页色调，框架、语言等提供了相应的改造办法。

外界的物体具有各种颜色，可以使物体显得鲜明和优美，使人产生不同的情感和爱好。过分鲜艳的颜色会使人产生倦怠的感觉，过分深谙的颜色则会使人的情绪感到沉重，红色和黄色可以给人一种

耀眼的感觉，青色和绿色给人带来凉爽和平静的感觉。各种颜色对光线的吸收和反射是各不相同的，红色对光线反射是67%，黄色反射是65%，绿色反射是47%，青色只反射36%。由于红色和黄色对光线反射比较强，因此容易产生耀光而刺眼。青色和绿色对光线的吸收和反射都比较适中，所以对人体的神经系统、大脑皮层和眼睛里的视网膜组织比较适应。因此，中国残联网的主要色调是采用绿色，如图12-16所示。

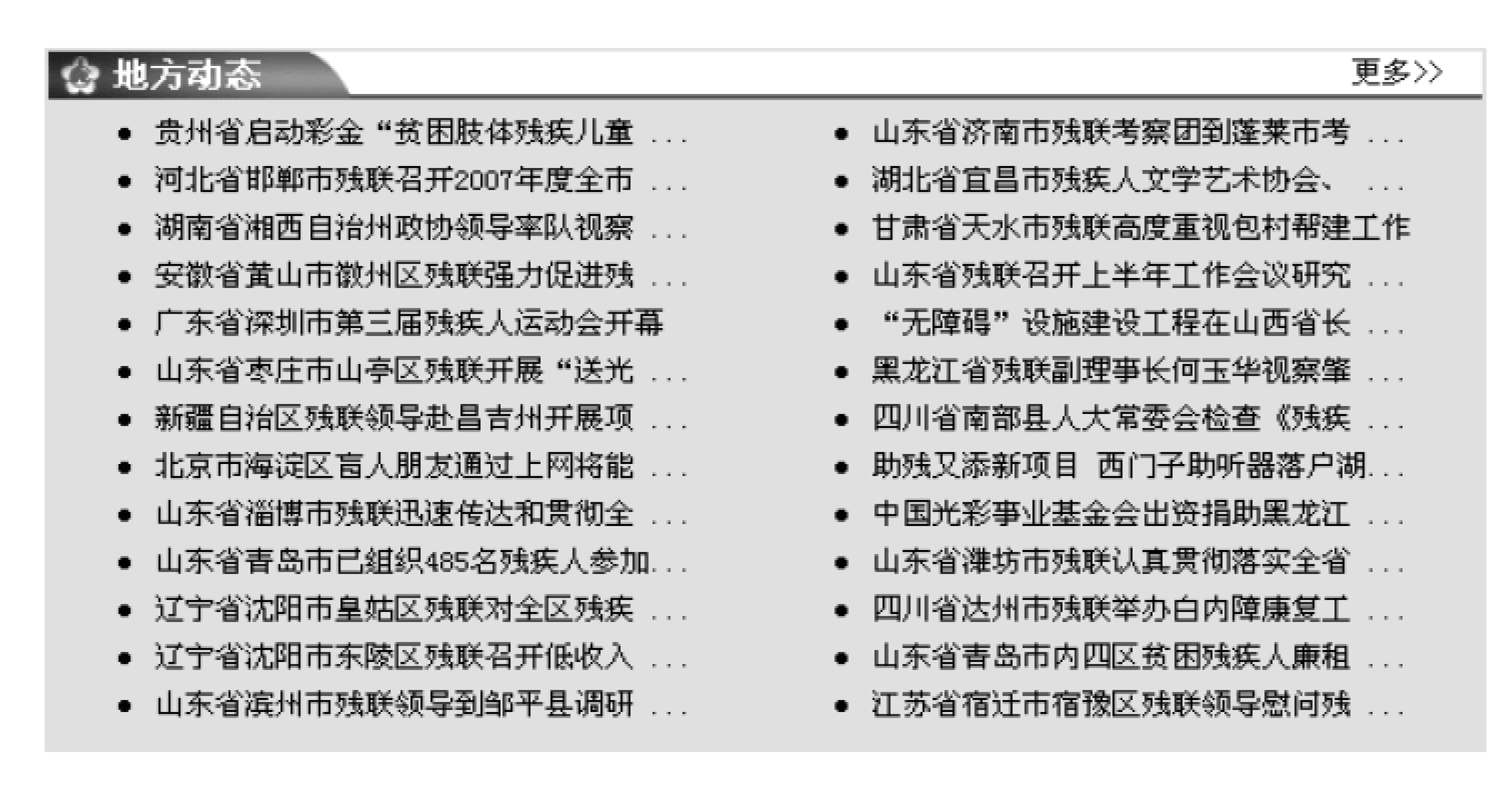


图12-16　网站主色调为绿色

就网页的框架结构而言，中国残联网分为四大区块：左侧选单区、主要选单区、右侧选单区和头部连接区。这种结构简单明了，有利于盲人、视力有障碍人群、认知能力有障碍的残疾人和老年人浏览网页。该四大区块都设定了快捷键，按下相应的快捷键就会到达相应的区块（注意左侧的小长方形），再继续按Tab键就可以到达该区块的相应内容。快捷键如下：

- Alt+C：左侧选单区，此区块呈现各网页的左侧内容。
- Alt+B：主要选单区，此区块呈现各网页的主要内容。
- Alt+N：右侧选单区，此区块呈现各网页的右侧内容。
- Alt+E：头部连接区，此区块列有本网站的首页头部连接。

以上设定快捷键的实现方法是通过使用accesskey指明键盘快捷键。

中国残联网还提供了三种版本：中文简体版、中文繁体版和英文版。不同的版本适用于不同的人士使用，以达到在语言方面的无障碍。其中，中文繁体版是通过看汉科技有限公司的繁简通HanWeb繁体系统实现的。

方式二，采用网站自动发声技术，提供语音版网站。

将网站中英文网页文字信息直接转换成标准流畅的语音文件，用户无须安装或下载任何软件便能通过浏览器听取网页内容，并可通过键盘数字键任意选择不同频道和栏目。该版本为盲人、视力有障碍人群、认知能力有障碍的残疾人和老年人提供了另外一种浏览网站的途径。

中国残疾人联合会盲人语音版网站是通过看汉科技有限公司的HanVoice网站自动发声系统搭建的。

方式三，提供网站浏览辅助工具（IBM Easy Web Browsing），适用于视力有障碍人

群、认知能力有障碍的残疾人和老年人,如图 12-17 所示。

① 自动朗读网页内容;

② 对网页的显示比例、文字及背景颜色进行定制;

③ 支持网站的多种语言版本;

④ 提供语速调整、音量调整、放大显示等个性化设置。

除了以上的功能外,还加入了很多其他的帮助功能。



图 12-17 IBM Easy Web Browsing

中国残联信息中心邀请了哈尔滨亿时代、IBM 公司(中国)、看汉科技有限公司进行无障碍合作,由 IBM 公司中国信息无障碍中心提供技术支持,由哈尔滨亿时代公司对中国残联网站进行重新设计,使用了看汉科技的技术。这里就采用的技术介绍如下:

① IBM Easy Web Browsing:是为了帮助视力不好或眼睛容易疲劳的人快速舒服地浏览网站而设计的软件,而且不需要复杂的安装。主要的功能包括:把鼠标移动到不容易浏览和阅读的地方,该处文字就会自动放大显示,自动用合成音朗读被放大显示的地方。

② HanWeb:是看汉科技有限公司推出的转换软件,可以对繁体、简体、Unicode 和广东话点字法进行转换。无须改变现有的网站,所有的维护和更新,包含内容、设计和程序都只需要一个版本。它是一个基于智能规则和数据库的引擎。繁简转译软件向企业内联网用户提供中文网络信息繁简转译。用户可在浏览中文网络信息时,自由选择以繁体或简体中文阅读。只需要在服务器上安装此软件,向用户提供简单、易用的使用接口。

③ HanVoice:也是看汉科技有限公司推出的软件,可以帮助使用个人计算机、移动电话、数据网络,浏览全球各种各样的信息。看汉及发声网站帮助用户通过用互联网和电话以广州话、普通话、英语三种语言形式,访问在线实时信息。通过电话方便地在互联网上查找和发布信息,浏览网站。同时也能帮助年老者、视障人士以及特殊人群得到信息。

就无障碍测试发现的很多问题,哈尔滨亿时代进行了重新编码和全面改版。值得介绍的是,中国残联网站声明遵循了 WCAG 2.0 国际标准,WCAG 2.0 尽管是尚未出台的正式标准,中残联可能考虑到要保持技术的先进性,保证未来几年内不过时,还是前瞻性地采用了此标准。

中国残联表示中国残联网站注重以人为本,倡导和实践信息无障碍理念,遵循国际标准,集合先进技术,为盲人、视力有障碍人群、认知能力有障碍的残疾人和老年人提供多种获取网上信息的方式,成为中国第一个能够以多种技术体现网站信息无障碍的门户网站。

2008年7月1日中国信息无障碍标准《信息无障碍身体机能差异人群网站设计无障碍技术要求》正式实施。中国互联网协会理事长胡启恒表示，这些网站的改造经技术小组测试，已初步达到了《信息无障碍—身体机能差异人群—网站设计无障碍技术要求》的一级标准。经过半年多的艰苦努力，北京2008奥运会、残奥会网站信息无障碍行动取得了阶段性的成果，成为网站信息无障碍的典型范例，必将推动我国信息无障碍的进一步推广和普及。

附录A UML 基础：统一建模语言简介

UML 是一种可视化的建模语言，结合了 Booch、Objectory 和 OMT 方法，同时吸收了其他方法学的一些思想，提供了一种表示的标准。1997 年，OMG 采纳 UML 作为软件建模语言的标准，可以应用于不同的软件开发过程。下面介绍最常用的 UML 图并给予简要说明，更详细的信息请参看《UML 参考手册》。初学者应该从了解 UML 的历史和原则起步。

A1 用 例 图

用例图描述了系统提供的一个功能单元。用例图的主要目的是帮助开发团队以一种可视化的方式理解系统的功能需求，包括基于基本流程的“角色”(actors，也就是与系统交互的其他实体)关系，以及系统内用例之间的关系。用例图一般表示出用例的组织关系，要么是整个系统的全部用例，要么是完成具有功能的一组用例，例如所有安全管理相关的用例。要在用例图上显示某个用例，可绘制一个椭圆，然后将用例的名称放在椭圆的中心或椭圆下面的中间位置。要在用例图上绘制一个角色(表示一个系统用户)，可绘制一个人形符号。角色和用例之间的关系使用简单的线段来描述，如图 A1 所示。

图 A1 中英文从上到下相应译为：CD 销售系统、查看乐队 CD 的销售统计、乐队经理、查看 Billboard 200 排行榜报告、唱片经理、查看特定 CD 的销售统计、检索最新的 Billboard 200 排行榜报告、排行榜报告服务。

用例图通常用于表达系统或者系统范畴的高级功能。从图 A1 中可以很容易看出该系统所提供的功能。这个系统允许乐队经理查看乐队 CD 的销售统计报告以及 Billboard 200 排行榜报告，也允许唱片经理查看特定 CD 的销售统计报告和这些 CD 在 Billboard 200 排行榜的报告。该图还告知系统将通过一个名为“排行榜报告服务”的外部系统提供 Billboard 排行榜报告。

此外，在用例图中，没有列出的用例表明了该系统不能完成的功能。例如，图 A1 不能提供给乐队经理收听 Billboard 200 上不同专辑中的歌曲的途径，也就是说，系统没有引用一个叫做“收听 Billboard 200 上的歌曲”的用例。在用例图中提供清楚的、简要的用例描述，项目赞助商就很容易看出系统是否提供了必须的功能。

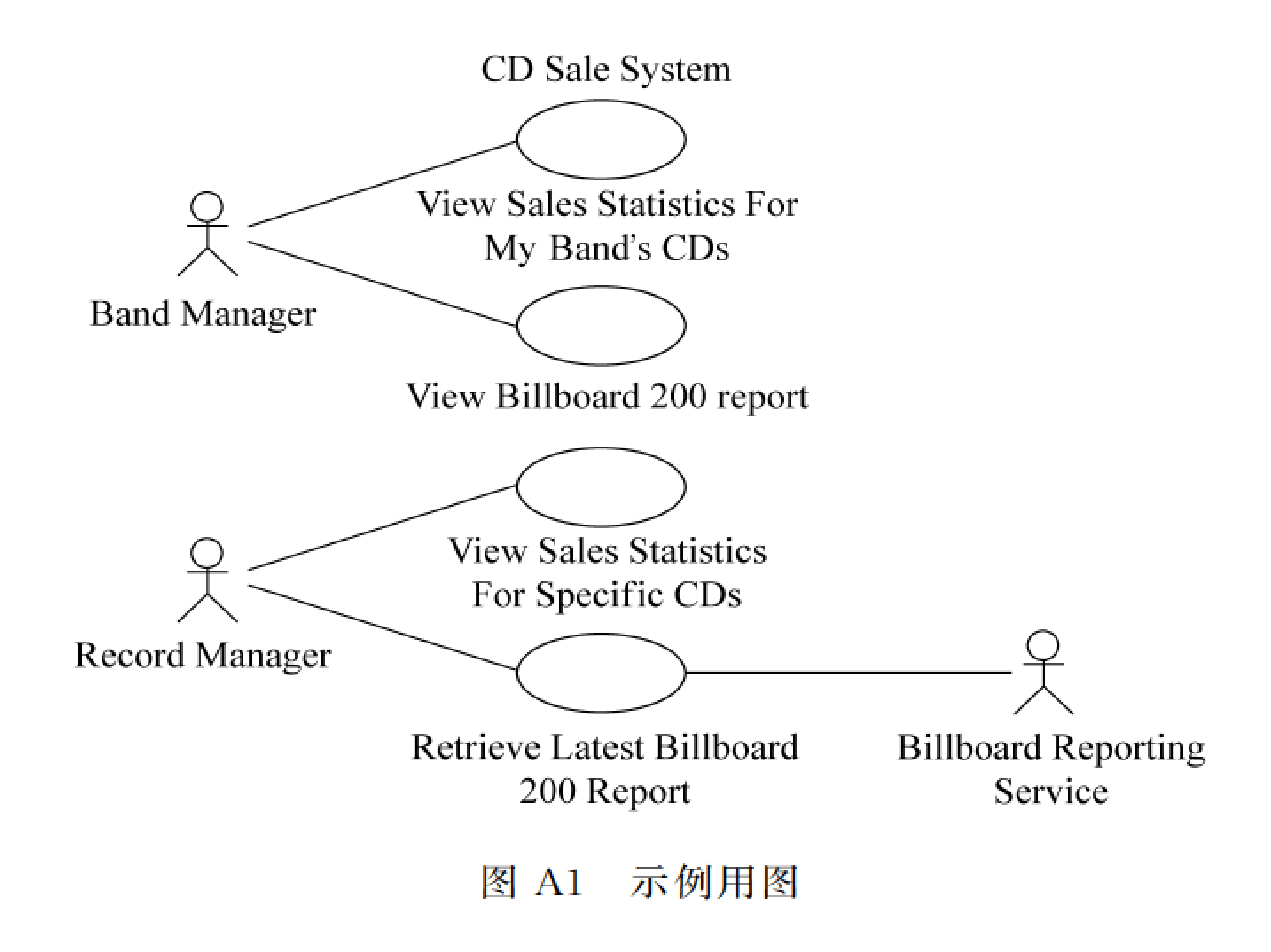


图 A1　示例用图

A2　类　　图

类是一种抽象,代表一组对象共有的结构和行为。类图表示类与类之间的关系以及类的属性和操作,换句话说,它描述了系统的静态结构。类图可用于表示逻辑类,通常指业务人员所谈及的事物种类,如摇滚乐队、CD、广播剧,或贷款、住房抵押、汽车信贷以及利率。类图还可用于表示实现类,实现类就是程序员处理的实体。实现类图或许会与逻辑类图显示一些相同的类。然而,实现类图不会使用相同的属性来描述,因为它很可能具有对诸如 Vector 和 HashMap 这种事物的引用。

类在类图上使用包含三个部分的矩形来描述,如图 A2 所示。最上面的部分显示类的名称,中间部分包含类的属性,最下面的部分包含类的操作(或方法)。

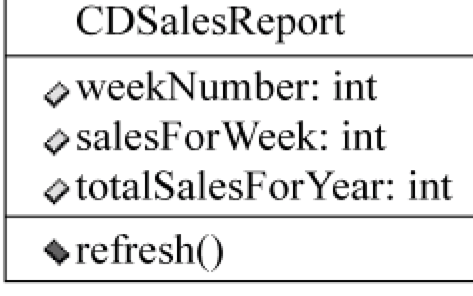


图 A2　类图中的示例类对象

在图 A3 中,包括了图 A2 所示的类对象。类图使用带有顶点指向父类的箭头的线段来绘制继承关系 1,并且箭头应该是一个完全的三角形。如果两个类都彼此知道对方,则应该使用实线来表示关联关系;如果只有其中一个类知道该关联关系,则使用开箭头表示。

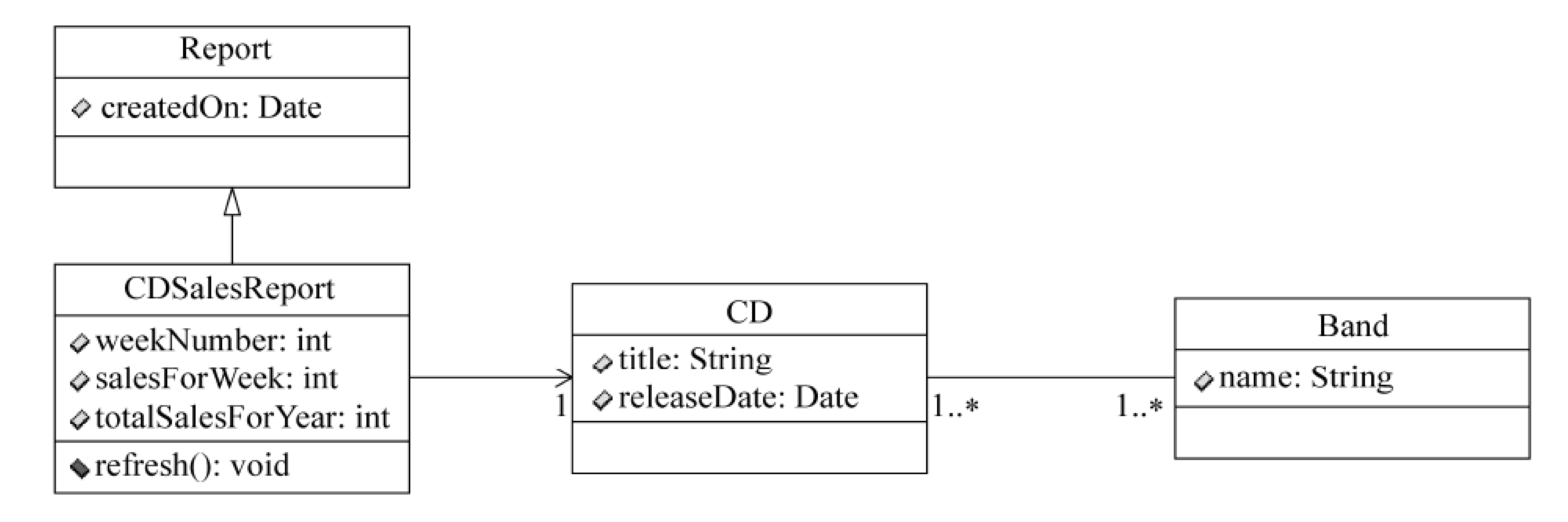


图 A3　一个完整的类图

在图 A3 中，同时看到了继承关系和两个关联关系。CDSalesReport 类继承自 Report 类。一个 CDSalesReport 类与一个 CD 类关联，但是 CD 类并不知道关于 CDSalesReport 类的任何信息。CD 类和 Band 类都彼此知道对方，两个类彼此都可以与一个或多个对方类相关联。

A3 序 列 图

序列图显示具体用例或用例的一部分的详细流程。它几乎是自描述的，并且显示了流程中不同对象之间的调用关系，同时还可以很详细地显示对不同对象的不同调用。

序列图有两个维度：垂直维度以发生的时间顺序显示消息/调用的序列，水平维度显示消息被发送到的对象实例。

序列图的绘制非常简单。横跨图的顶部，每个框(见图 A4)表示每个类的实例(对象)。在框中，类实例名称和类名称之间用空格/冒号/空格来分隔，如 myReportGenerator：ReportGenerator。如果某个类实例向另一个类实例发送一条消息，则绘制一条具有指向接收类实例的开箭头的连线，并把消息/方法的名称放在连线上面。对于某些特别重要的消息，可以绘制一条具有指向发起类实例的开箭头的虚线，将返回值标注在虚线上。其实如果还绘制出包括返回值的虚线，这些额外的信息可以使得序列图更易于阅读。

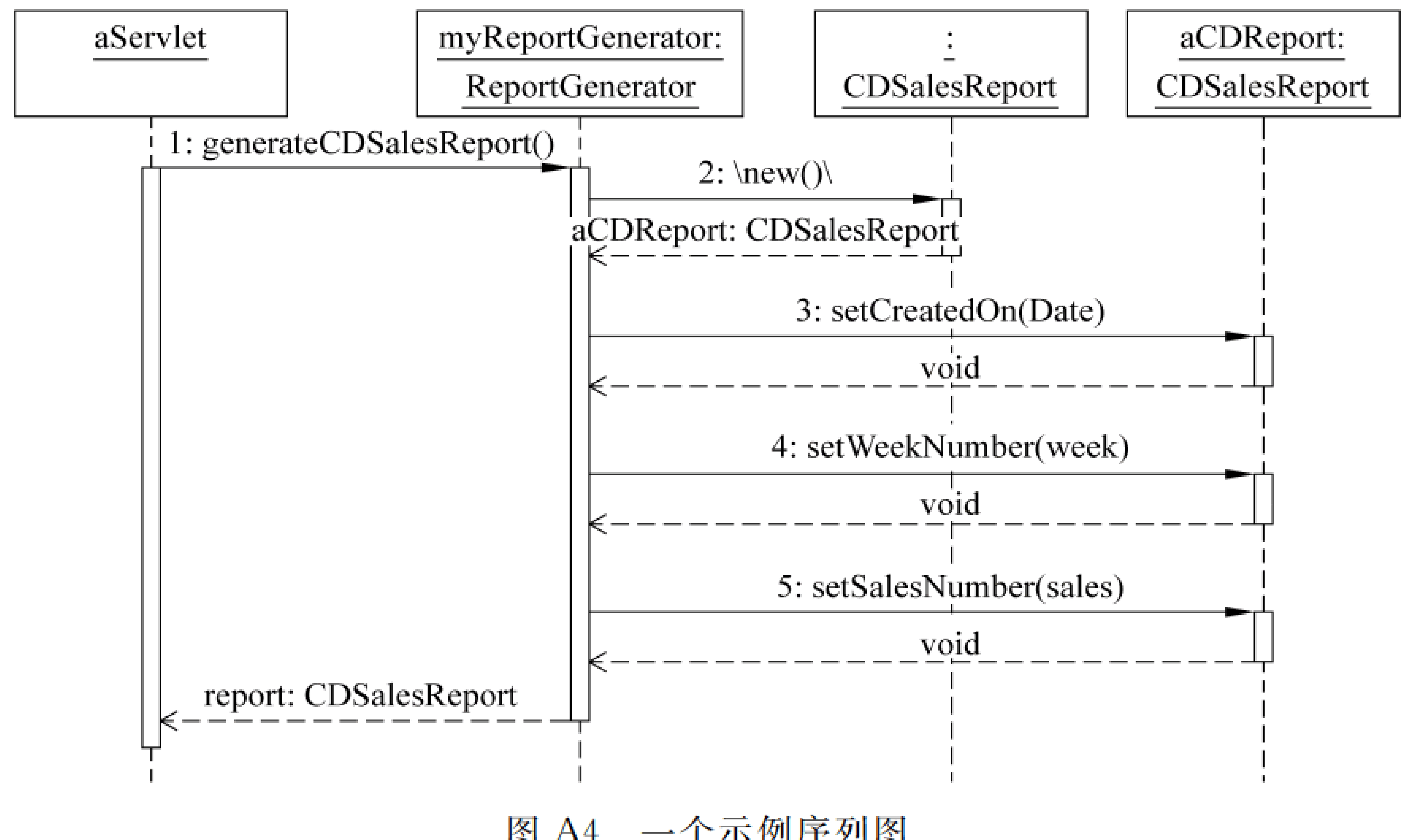


图 A4 一个示例序列图

阅读序列图也非常简单。从左上角启动序列的“驱动”类实例开始，然后顺着每条消息往下阅读。虽然图 A4 所示的例子序列图显示了每条被发送消息的返回消息，但这只是可选的。

通过阅读图 A4 中的示例序列，可以明白如何创建一个 CD 销售报告(CD Sales Report)。其中的 aServlet 对象表示驱动类实例。aServlet 向名为 gen 的 ReportGenerator 类实例发送一条消息。该消息被标为 generateCDSalesReport，表示 ReportGenerator 对象

实现了这个消息处理程序。进一步理解可发现，generateCDSalesReport 消息标签在括号中包括了一个 cdId，表明 aServlet 随该消息传递一个名为 cdId 的参数。当 gen 实例接收到一条 generateCDSalesReport 消息时，它会接着调用 CDSalesReport 类，并返回一个 aCDReport 的实例。然后 gen 实例对返回的 aCDReport 实例进行调用，在每次消息调用时向它传递参数。在该序列的结尾，gen 实例向它的调用者 aServlet 返回一个 aCDReport。

注意：图 A4 中的序列图相对于那些典型的序列图来说比较详细，它更易于理解，并且它显示了如何表示嵌套的调用。对于初级开发人员来说，有时把一个序列分解到这种详细程度是很有必要的，这有助于理解相关的内容。

A4 状 态 图

状态图表示某个类所处的不同状态和该类的状态转换信息。有人可能会说每个类都有状态，但不是每个类都应该有一个状态图。对“感兴趣的”状态的类，实际上是说在系统活动期间具有三个或更多潜在状态的类才进行状态图描述。

如图 A5 所示，状态图的符号集包括 5 个基本元素：初始起点，它使用实心圆来绘制；状态之间的转换，它使用具有开箭头的线段来绘制；状态，它使用圆角矩形来绘制；判断点，它使用空心圆来绘制；一个或者多个终止点，它们使用内部包含实心圆的圆来绘制。要绘制状态图，首先绘制起点和一条指向该类的初始状态的转换线段。状态本身可以在图上的任意位置绘制，然后只需使用状态转换线条将它们连接起来。

图 A5 中的状态图显示了它们可以表达的一些潜在信息，从中可以看出贷款处理系统最初处于 Loan Application 状态。当批准前(pre-approval)过程完成时，根据该过程的

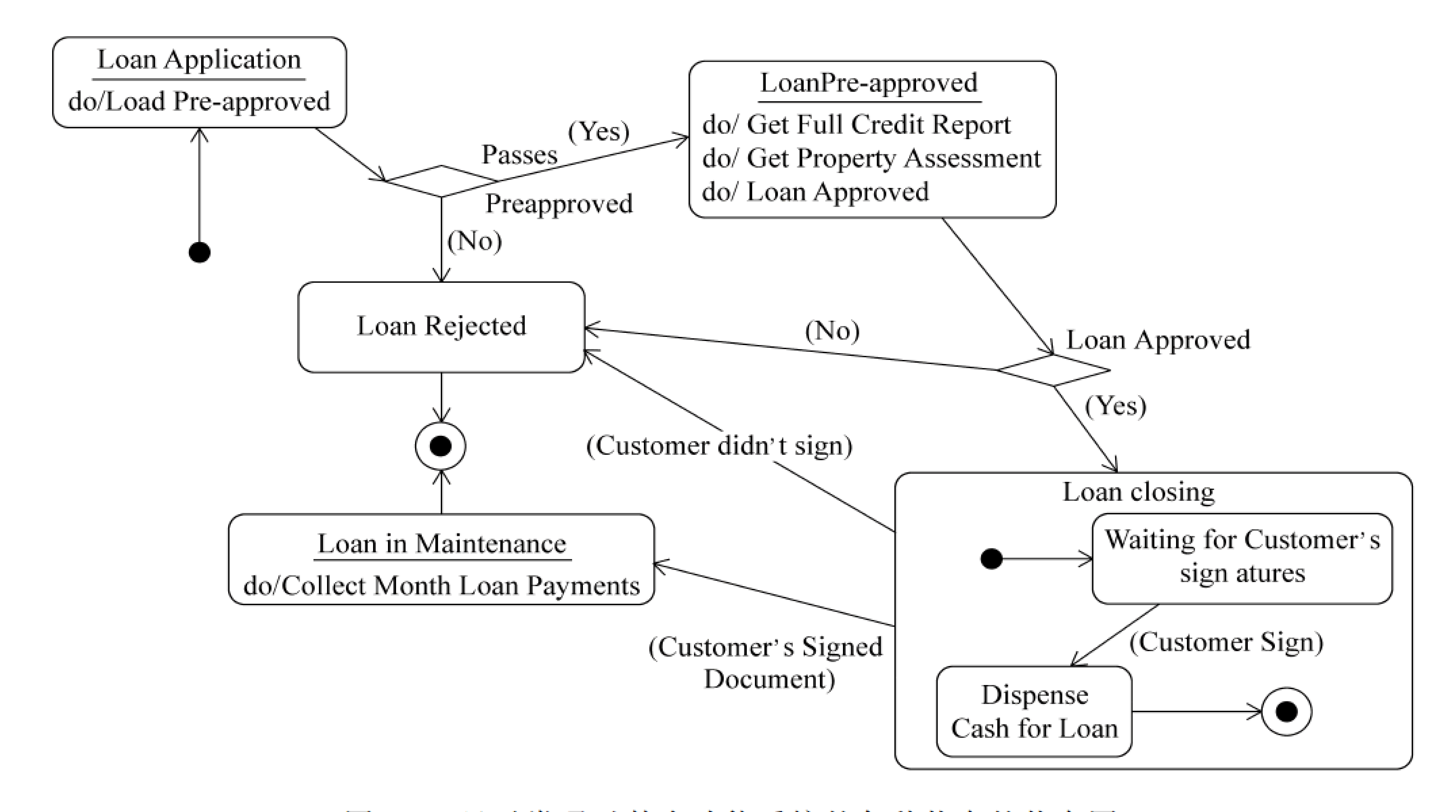


图 A5　显示类通过某个功能系统的各种状态的状态图

结果,或者转到 Loan Pre-approved 状态,或者转到 Loan Rejected 状态。这个判断(它是在转换过程期间做出的)使用一个判断点来表示即转换线条间的空心圆。通过该状态图可知,如果没有经过 Loan Closing 状态,贷款不可能从 Loan Pre-Approved 状态进入 Loan in Maintenance 状态。而且,所有贷款都将结束于 Loan Rejected 或 Loan in Maintenance 状态。

A5 活 动 图

活动图表示在处理某个活动时两个或者更多类对象之间的过程控制流。活动图可用于在业务单元的级别上对更高级别的业务过程进行建模,或者对低级别的内部类操作进行建模。活动图最适合用于对较高级别的过程建模,例如公司当前在如何运作业务。这是因为与序列图相比,活动图在表示上“不够技术性的”,但有业务头脑的人们往往能够更快速地理解它们。

活动图的符号集与状态图中使用的符号集类似。像状态图一样,活动图也从一个连接到初始活动的实心圆开始。活动是通过一个圆角矩形(活动的名称包含在其内)来表示的。活动可以通过转换线段连接到其他活动,或连接到判断点,这些判断点连接到由判断点的条件所保护的不同活动。结束过程的活动连接到一个终止点(就像在状态图中一样)。作为一种选择,活动可以分组为泳道(swimlane),泳道用于表示实际执行活动的对象,如图 A6 所示。

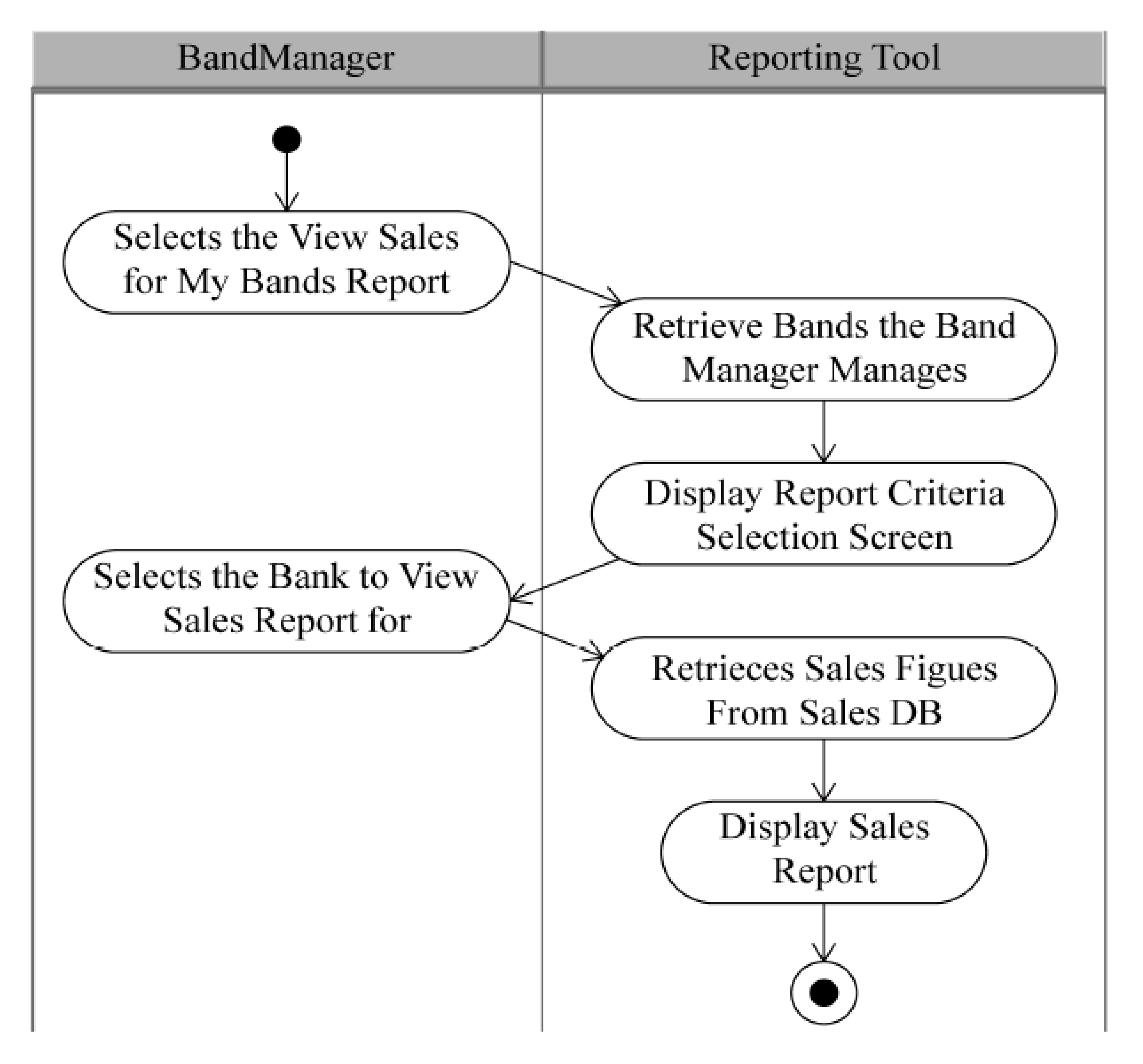


图 A6 活动图(具有两个泳道,表示两个对象的活动控制:乐队经理和报告工具)

在图 A6 中,英文相应译为:乐队经理、报告工具、选择查看乐队的销售报告、检索该乐队经理所管理的乐队、显示报告条件选择屏幕、选择要查看其销售报告的乐队、从销售

数据库检索销售数据、显示销售报告。

该活动图中有两个泳道,因为有两个对象控制着各自的活动:乐队经理和报告工具。整个过程首先从乐队经理选择查看他的乐队销售报告开始,然后报告工具检索并显示他管理的所有乐队,并要求他从中选择一个乐队。在乐队经理选择一个乐队之后,报告工具就检索销售信息并显示销售报告。该活动图表明,显示报告是整个过程中的最后一步。

A6 组 件 图

组件图提供系统的物理视图。它的用途是显示系统中的软件对其他软件组件(如库函数)的依赖关系。组件图可以在一个非常高的层次上显示,从而仅显示粗粒度的组件,也可以在组件包层次 2 上显示。

组件图的建模最适合通过例子来描述。图 A7 显示了 4 个组件:Reporting Tool、Billboard Service、Servlet 2.2 API 和 JDBC API。从 Reporting Tool 组件指向 Billboard Service、Servlet 2.2 API 和 JDBC API 组件的带箭头的线段,表示 Reporting Tool 依赖于那三个组件。

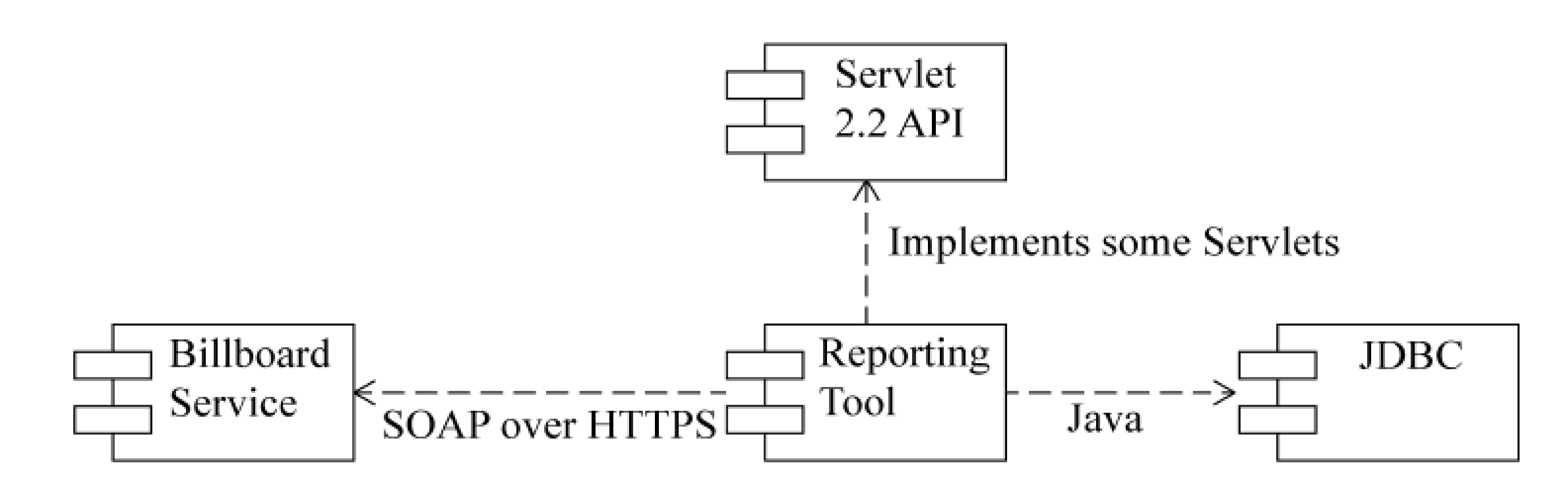


图 A7　组件图显示了系统中各种软件组件的依赖关系

A7 部 署 图

部署图表示该软件系统如何部署到硬件环境中,用途是显示该系统不同的组件将在何处物理地运行,以及它们将如何彼此通信。因为部署图是对物理运行情况进行建模,系统的生产人员就可以很好地利用这种图。

部署图中的符号包括组件图中所使用的符号元素,另外还增加了几个符号,包括节点的概念。一个节点可以代表一台物理机器,或代表一个虚拟机器节点(如一个大型机节点)。要对节点进行建模,只需要绘制一个三维立方体,节点的名称位于立方体的顶部,所使用的命名约定与序列图中相同:[实例名称]:[实例类型](例如,w3reporting. myco. com:Application Server)。

由于 Reporting Tool 组件绘制在 IBM WebSphere 内部,后者又绘制在节点 w3. reporting. myco. com 内部,因而用户将通过运行在本地机器上的浏览器来访问 Reporting Tool,浏览器通过公司 Intranet 上的 HTTP 协议与 Reporting Tool 建立连接。

图 A8 中的部署图表明,用户使用运行在本地机器上的浏览器访问 Reporting Tool,

并通过公司 Intranet 上的 HTTP 协议连接到 Reporting Tool 组件。这个工具实际运行在名为 w3reporting. myco. com 的 Application Server 上。这个图还表明 Reporting Tool 组件绘制在 IBM WebSphere 内部，后者又绘制在 w3. reporting. myco. com 节点内部。Reporting Tool 使用 Java 语言通过 IBM DB2 数据库的 JDBC 接口连接到它的报告数据库上，然后该接口又使用本地 DB2 通信方式，与运行在名为 db1. myco. com 的服务器上实际的 DB2 数据库通信。除了与报告数据库通信外，Report Tool 组件还通过 HTTPS 上的 SOAP 与 Billboard Service 进行通信。

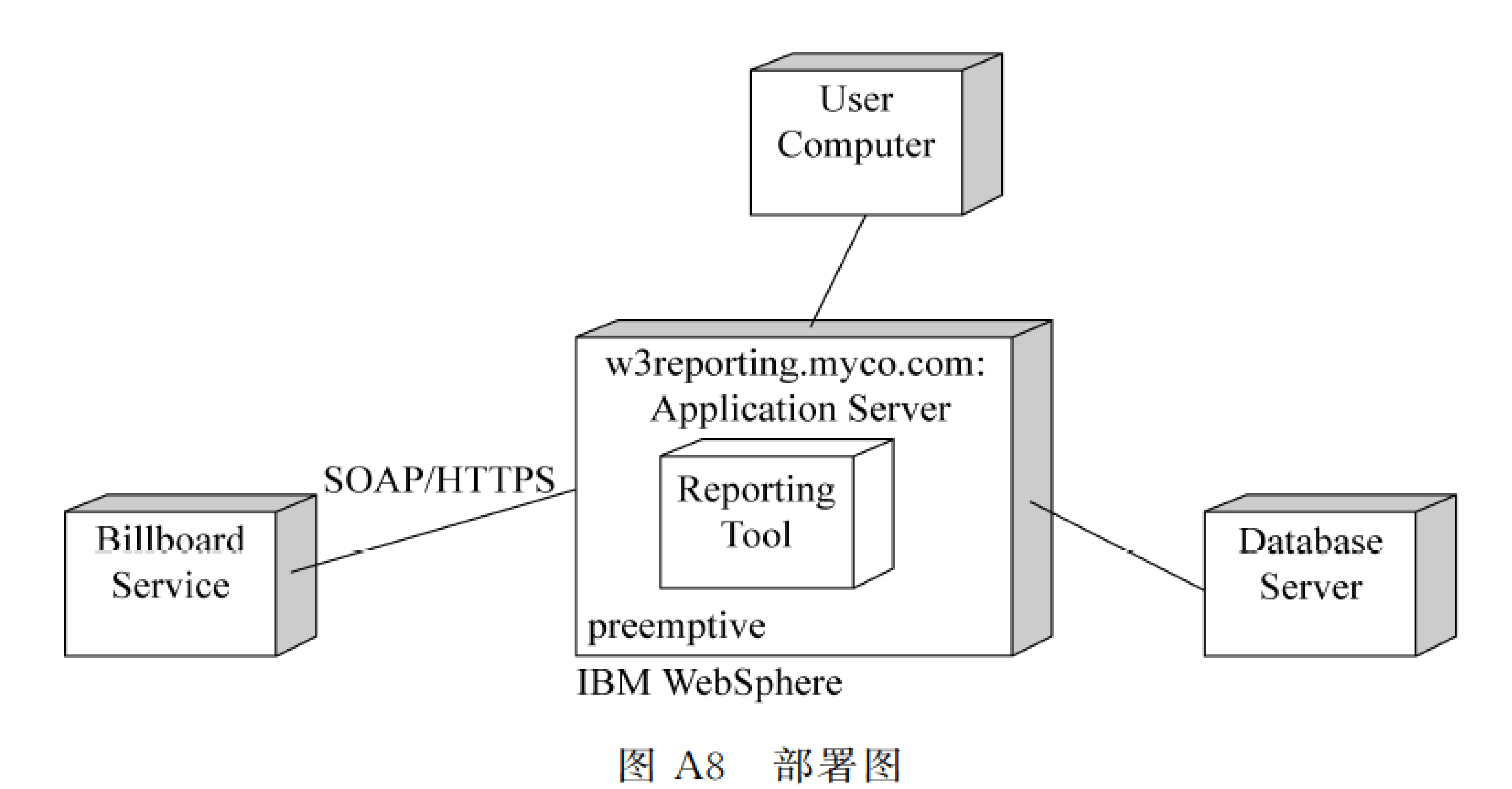


图 A8　部署图

附录 B 测试评估摘要

说明：RUP 提供了详细生动的工作指南和文档模板，此模板为其中之一，仅供参考，读者可参阅更多的模板。RUP 模板更多的是关注于过程的管理，提供一个稳定的、具有指导作用的框架，而不是一些具体的、涉及操作细节的方法。

＜公司名称＞

＜项目名称＞

测试评估摘要

版本＜1.0＞

［注：以下提供的模板用于 Rational Unified Process。其中包括用方括号括起来并以斜体(样式＝InfoBlue)显示的文本，它们用于向作者提供指导，在发布此文档之前应该将其删除。按此样式输入的段落将被自动设置为普通样式(样式＝Body Text)。］

［要定制 Microsoft Word 中的自动字段(选中时显示灰色背景)，请选择 File＞Properties，然后将 Title、Subject 和 Company 等字段替换为此文档的相应信息。关闭该对话框后，通过选择 Edit＞Select All(或 Ctrl＋A)并按 F9 键，或只是在字段上单击并按 F9 键，可以在整个文档中自动更新字段。对于页眉和页脚，这一操作必须单独进行。按 Alt＋F9 键，将在显示字段名称和字段内容之间切换。有关字段处理的详细信息，请参见 Word 帮助。］

＜项目名称＞	Version：＜1.0＞
测试评估摘要	Date：＜dd/mmm/yy＞
＜document identifier＞	

修订历史记录

日　期	版　本	说　明	作　者
＜日/月/年＞	＜x.x＞	＜详细信息＞	＜姓名＞

Confidential　　@＜公司名称＞，2000　　Page 2 of 4

目　　录

测试评估摘要

1. 简介

[测试评估摘要的简介应提供整个文档的概述。它应包括此测试评估摘要的目的、范围、定义、首字母缩写词、缩略语、参考资料和概述。]

1.1 目的

[阐明此测试评估摘要的目的。]

1.2 范围

[简要说明此测试评估摘要的范围:它的相关项目,以及受到此文档影响的任何其他事物。]

1.3 定义、首字母缩写词和缩略语

[本小节应提供正确理解此测试评估摘要所需的全部术语、首字母缩写词和缩略语的定义。这些信息可以通过引用项目词汇表来提供。]

1.4 参考资料

[本小节应完整列出此测试评估摘要文档中其他部分所引用的任何文档。每个文档应标有标题、报告号(如果适用)、日期和出版单位。列出可从中获取这些参考资料的来源。这些信息可以通过引用附录或其他文档来提供。]

1.5 概述

[本小节应说明此测试评估摘要中其他部分所包含的内容,并解释此文档的组织方式。]

2. 测试结果摘要

[简要地总结测试的结果。]

3. 基于需求的测试覆盖

[对于您已选择使用的各种评测,指出其结果。与以前的结果进行比较,并讨论变化趋势。]

4. 基于代码的测试覆盖

[对于您已选择使用的各种评测,指出其结果。与以前的结果进行比较,并讨论变化趋势。]

5. 建议措施

[根据对测试结果和主要测试评测结果所进行的评估,建议任何可取的措施。]

6. 图

[提供测试结果或主要测试评测结果的所有图形表示。]

附录 C WCAG 1.0 的 14 条指导原则

原则 1：为网页的音频和视频信息提供等价物代替。

这条原则要求在设计网页内容时，必须考虑到一些特殊用户，他们可能无法使用图像文件、电影文件、声音文件以及程序插件等，但是必须为他们提供等价物，从而可以顺利理解页面的内容。例如，可以使用一些文本或手势作为图像（标注）和视频文件（字幕、手语）的等价物，使得聋人通过文本或手势可以获得听觉信息；也可以通过在网页内容中插入音频剪辑作为视频信息的等价物，从而使盲人获得视觉信息。由于文本内容最容易通过各种方式和各种形式呈现给各类残疾人群，因此该原则还重点强调了使用替换文字的重要性，即预先生成各种非文本内容（图表、图像、电影、声音等）的替换文字。文本内容可以被合成语音或被翻译为盲文，或者在屏幕、纸上以不同尺寸显示。合成语音对于盲人、有阅读障碍或理解和学习障碍的人非常重要。盲文有利于那些失去视觉的用户理解其内容。具体方法是为所有非文本内容提供替换文字；为服务器端网站地图所包含每个有效区域提供额外的文字链接；提供有关听觉上的描述内容，以表达视频或多媒体呈现的重点信息；对于任何时间性的多媒体内容提供播放同步化的等价物。

原则 2：不能只依靠色彩来提供信息。

这条原则要求网页内容在没有色彩的情况下，确保文本和图像仍然是可以理解的。如果只是依靠色彩来传递信息，那么那些无法辨别色彩和只有单色显示器或无显示器的用户就无法获得信息。当网页的前景色和背景色过于接近的时候，就不能通过一定的色彩对比度来为那些有视觉缺陷的人提供信息。具体方法是保证所有通过色彩来传递信息的网页内容在无色情况下也能有效传递；保证网页内容前景色与背景色具有足够的对比度，从而保证在色盲患者眼中或在黑白显示器中也能分辨出来。

原则 3：使用合适的标记语言和样式表。

这条原则要求使用结构化元素来标记整个网页文档，使用样式表 CSS（Cascading Style Sheets）而非表现元素和属性来控制网页的表现形式。不规范合理的标记和滥用表现层的标记会给用特定软件访问的用户造成理解和导航上的困难。为了适应某些旧的网页浏览器，内容开发者可能会使用一些标记来达到某种想要的设计格式或特效，但是这样做可能会对网页的访问产生很大的阻碍，要权衡考虑采用某种特定页面设计是否比保证所有人获取文档信息更重要。另一方面，内容开发者也不要因为有一部分浏览器和辅助技术无法正确处理，而把这些标签给供奉起来。例如说，在 HTML 里，对于表格状信息

应该使用 TABLE 元素来标记，即使还有一些旧的屏幕阅读器无法正确处理那些相邻的文本。具体方法是使用已发布的正式语法来创建文档；使用样式表来控制布局和表现；在标记语言的属性值和样式表的属性值中，尽量使用相对单位，而非绝对单位；根据规范使用标题呈现文档结构；避免利用标记引用语来制造缩排等排版效果。

原则 4：明确自然语言的使用方法。

这条原则要求使用标记来进行对缩写文字和外国文字的拼写和解释。当内容开发者为网页文档中的自然语言变换作了标记，语音合成器和盲文设备会自动切换至新的语言，让多语言使用者可以顺利访问文档。内容开发者应该通过使用标记或 HTTP 头来确定文档内容使用的主要自然语言，还需要为缩写和简称提供解释说明。除了有利于提供技术上的帮助，自然语言标记也使得搜索引擎可以更好地获取指定语言的关键字。自然语言标记为所有人群提高了网页的可读性，包括有学习障碍和认知障碍的人群以及聋人。具体方法是文档中任何文字所使用的自然语言更换时，给予明确的识别；文档中缩写文字或简称第一次出现时，应注明全称。

原则 5：创建结构良好的表格。

这条原则要求确保网页表格具备良好的结构，从而能够被浏览器和其他用户代理顺利呈现。表格是用来呈现真正的数据信息的，内容开发者应当尽量避免使用表格来建构整个页面，因为表格页面在用户通过听觉访问网页（使用屏幕阅读器或自动计算机）时，或者每次只浏览一部分网页内容（使用语音输出或盲文显示）时都会出现很多问题。具体方法是：对于数据表格，明确行和列的标题；对于具有两层或多层行列逻辑关系的表格，合理使用标签明确单元格与标题的关系；如果已经使用表格来布局，就不能使用其他的结构性标记来处理视觉格式效果。

原则 6：保证页面在新技术条件下的良好呈现。

这条原则要求网页内容即使在不支持新技术的条件下也能被用户顺利访问。尽管我们提倡使用新技术，解决现存技术中存在的一些问题，但应该保证那些使用旧版浏览器或关闭了新技术的用户能够顺利访问网页内容。具体方法是组织文档，使其在没有样式表的情况下也能阅读；保证动态内容与替换文字能够一致更新；保证在脚本、小应用程序或其他程序型对象关闭或不支持的情况下，网页仍可使用；保证事件处理程序应与输入接口及设备无关。

原则 7：保证用户可以控制时间敏感性内容的变化。

这条原则要求网页中那些移动的、闪烁的、自动更新的对象或页面能够被关闭或暂停。一些视力障碍和认知障碍的用户可能无法阅读快速移动的文字，而且快速移动的文字还会干扰用户对其他文字的阅读。屏幕阅读器无法读出快速移动的文字，肢体残疾的用户也无法跟踪快速移动的对象，从而产生理解上的困难。具体方法是应尽量避免屏幕和网页内容闪烁，避免页面内出现移动内容；不要创建周期性自动刷新的页面，尽量避免页面自动重新定向，而是通过服务器来实现重新定向。

原则 8：保证嵌入式用户界面的无障碍性。

这条原则要求保证用户界面遵循无障碍设计原则，用户可以使用键盘操作及自动发音系统，而与功能操作设备没有直接联系。当一个嵌入式对象具备“自己的界面”，而且和

浏览器的界面类似，那么对于用户来说是无障碍的。如果嵌入式对象的界面不能达到此要求，那么最好有一个可替代的无障碍方案。具体方法是让程序型元素，如脚本和小应用程序等，能够直接被一些辅助技术使用，或与之兼容。

原则 9：设备无关性的设计。

这条原则要求保证各种不同的输入设备都能激活页面元素，即应用各种输入设备都可以无障碍地访问网页内容。设备无关性意味着用户可以使用一些自己喜欢的输入（或输出）设备（如鼠标、键盘、语音、肢体棒等）来和用户代理或文档进行交互。如果有些页面只适合采用某些特定的设备访问，那么某些残疾人士就无法正常访问页面了。例如，表单内容如果只能通过鼠标激活，那么视力障碍人群和使用语音输入或键盘输入的人群就无法访问了。具体方法是提供用户端的图像映射，而非服务器端；保证任何具有自身操作界面的元素，其操作方式都与使用者的设备无关；指定有关脚本的逻辑上的事件处理，而不是特定装置的事件处理；为重要的链接提供键盘快捷键方式。

原则 10：使用过渡性解决方案。

这条原则要求使用过渡性的网页访问无障碍解决方案，从而使那些计算机辅助设备和旧版浏览器能够正常运行。例如，旧版浏览器一般不允许用户浏览空的编辑框，旧版屏幕阅读器通常把连续的链接列表理解为一个链接，类似的这些页面活动元素访问起来比较困难或难以访问。另外，改变当前窗口内容或者弹出新窗口则会使一些用户感到茫然。具体方法是尽量避免使用弹出式窗口或其他类似窗口，避免在未通知用户的情况下就变更当前窗口；确保这些标签位于合适的位置，以保证与关联的表单控制元素的联系；在编辑框及文字区域中预先放置占位字符以正确处理空白的控制元素；在两个链接间插入不属于该链接又可被打印的字符（并以空格隔开）以清楚显示紧靠的两个链接。

原则 11：使用 W3C 的技术和指南。

这条原则要求使用 W3C 技术规范并且遵循无障碍性指南。如果使用 W3C 技术有困难，或可能会造成内容呈现上的问题，那么可以提供一个现有内容的无障碍性替代版本。由于 W3C 技术本身遵循无障碍性规范，并且是开放和标准的，可以避免因使用非 W3C 推荐和非标准的功能所造成的障碍，可以使更多的人使用多种软硬件访问页面。很多非 W3C 格式（如 PDF，Shockwave 等）必须安装插件或者独立的应用程序来浏览。通常情况下这些格式不能被标准的用户代理（包括相关辅助技术）阅读和导航，因此必须提供等价的替代页面。当然，使用 W3C 技术也必须遵循无障碍性指南，当使用新技术时要保证页面内容能够良好地呈现出来。具体方法是尽可能合理地使用被支持的最新的 W3C 技术，避免使用 W3C 废弃的功能。如果实在无法创建无障碍性网页，应另外提供使用 W3C 推荐的技术，具备可访问性的网页，并且提供信息或功能的等价物，保持和原网页的同步更新。

原则 12：提供内容引导信息。

这条原则要求提供网页上下文和引导信息，以帮助用户理解复杂的页面或元素。提供相关页面间和相关元素间关系的信息，可以帮助所有的用户理解。有些页面间复杂的联系可能会造成认知障碍人士和视觉障碍人士访问上的困难。具体方法是为每一个框架添加标题，以促进框架的辨认与导航，或者把大块的信息分隔为易于管理的小部分，并插

入标签元素。

原则 13：提供清晰的导航机制。

这条原则要求提供清晰并且一致的导航机制，包括引导信息、导航条、网站地图等，从而为用户在网站中迅速而精确地找到特定信息提供便利。清晰并且统一的导航机制不仅对于具有认知障碍、视力障碍的人非常重要，而且对于所有人都是有利的。具体方法是在页面和网站中加入语义化信息，并提供网站结构规划方面的信息（如网站地图或者目录索引）。相关的链接文字无论是单独理解或是放入上下文中都应该是意义清楚、简明扼要的，且内容开发者应该利用信息提示指明链接的目标。另外，网页开发者还有必要提供搜索功能，并设计不同的网页内容搜索方式，让不同经验与喜好者选择使用。

原则 14：保证文档内容的清晰和简单。

这条原则要求提供清晰和简单的文档内容，以便于理解。统一的页面布局便于辨别的图片（替换文字）以及易于理解的文字对于所有用户，尤其是有认知障碍和阅读障碍的人都是非常有益的。对于认知障碍和学习障碍人士来说，较为复杂的文字会造成理解上的困难，而使用清晰简单的文字可以提高信息交流的效率。使用清晰和简单的语言对于那些母语并非网页语言的访问者和主要使用手语的人同样也是有益的。具体方法主要有使用最清晰简单的文字表达网站的内容，并提供图片、音频表达的文字补充说明，从而增强页面内容的易理解性，另外还需要统一页面之间的表现样式。

参 考 文 献

[1] Ron Patton.软件测试.第 2 版.北京：机械工业出版社，2006.
[2] 陈少英等.Web 性能测试实战.北京：电子工业出版社，2006.
[3] 何萍.自动化测试框架的设计与实现，南京大学硕士论文.2004.
[4] 贺平.软件测试技术.北京：机械工业出版社，2004.
[5] 李杰，青光辉，胡谷雨.基于脚本的自动化软件测试，2004 全国软件与应用学术会议（NASAC）.2004.9：197～201.
[6] 吴斌.软件功能自动化测试的探讨与应用，06MIs/S&A 学术交流会.2006.5：158～162.
[7] 杨文宏，李心辉等译.面向对象的软件测试.北京：中信出版社，2002.
[8] 冯玉刁，唐艳，周淳.关键字驱动自动化测试的原理与实现.计算机应用.2004.8：140～142.
[9] 赵瑞莲.软件测试.北京：高等教育出版社，2004.
[10] Gary Pollice 等著.宋锐等译.小型团队软件开发：以 RUP 为中心的方法.北京：中国电力出版社，2004.
[11] Mark Fewster，Dorothy Graham.软件测试自动化技术与实例详解.北京：电子工业出版社，2000.
[12] 朱少民.软件测试方法和技术.北京：清华大学出版社，2005.
[13] 罗运模等.软件能力成熟度模型集成（CMMI）.北京：清华大学出版社，2003.
[14] 飞思科技产品研发中心.实用软件测试方法与应用.北京：电子工业出版社，2003.
[15] 郭荷清等.现代软件工程——原理、方法和管理.广州：华南理工大学出版社，2004.
[16] 一起测试网，http：//www.17testing.com/.
[17] 51Testing 软件测试网，http：//www.51testing.com/.
[18] 中国软件测试时代，http：//www.testage.net/.
[19] 中国测试员网站，http：//www.cntester.com/index.shtml/.
[20] 点击测试网，http：//www.nsclick.com/web/.
[21] 易测网，http：//www.easytesting.cn/.
[22] 测试中国，http：//www.testingcn.com/html/index.html.
[23] 北大测试主站，http：//www.btesting.com/index.asp.
[24] 中国软件质量网，http：//www.rjzl.gov.cn/.
[25] 中国软件评测中心，http：//www.cstc.org.cn/.
[26] 赛宝软件评测中心，http：//www.scstlab.com.cn/.
[27] oldsidney 学习笔记，http：//www.oldsidney.idv.tw/.
[28] Humphrey W. A. Discip line for Software Engineering. Addison2Wesley Publishing Company. 1997.
[29] GradyBooch，Ivar Jacobson，James Rumbaugh. UnifiedModeling Language 1. 3，White paper [M]. New York：Rational Software Corp. 1998.
[30] Per Kroll 等著.徐正生等译.Rational 统一过程：实践者指南.北京：中国电力出版社，2004.
[31] RandyMaseiana.利用 IBM Rational Funetional Tester 6.1 实现可复用的测试框架，http：//www.ibm.eodeveloperwos/erational/419.
[32] 克鲁奇特，麻志毅等.RUP 导论.第 3 版.北京：机械工业出版社，2004.
[33] Ivar Jacoboson，Grady Booch，James Rumbaugh，The Unified Software Development Process. Addision Wesly Longman，Inc. 1999.
[34] James Rumbaugh，Ivar Jacoboson，Grady Booch，The Unified Modeling Language Reference Manual. Addision Wesly Longman，Inc. 1999.
[35] Grady Booch，James Rumbaugh，Ivar Jacoboson，Grady Booch，The Unified Modeling Language

User Guide. Addision Wesly Longman, Inc. 1999.
[36] Cockburn A. Writing Effective Use Cases. Addision Wesly Longman, Inc. 2000.
[37] The Unified Modeling Language Reference Manual, James Rumbaugh, Ivar Jacobson, Grady Booch, OMG.
[38] OMG Unified Modeling Language Specification V1. 3, OMG, 1999. 6.
[39] Rational Unified Process: Best Practices For Software Development Teams, Rational Software Corp. White Paper.
[40] Unifying Enterprise Development Teams With UML, Grady Booch, Rational Software White Paper.
[41] Rational Software Corporation. Rational Unified Process version 2000. 02. 1, 2000.
[42] Ivar Jacobson, Grady Booch, James Rumbaugh. The Unified Software Development Process, Addison Wesley, 1999. 1.
[43] Scott W. Ambler. Enhancing the Unified Process: Software Process for Large Scale, Mission-Critical Systems A Ronin Internatinal White Paper, 2000. 9.
[44] StePhen H. Kan. Metriesand Modelsin Software Quality Engineering (2ndEdition). Addison-Wesley Professional, 2002. 9.
[45] Glenford J. Myers. The Art of Software Testing, Second Edition. JolmWiley&Sons, Ine, 2004. 7.
[46] Damoel J. Mosley, BrueeA. Posey. Just Enough Software Test Automation. Pearson Education, Inc. , 2002.
[47] Dustin E. Automated Testing Lifeeyele Methodology (ATLM). STAREAST 2000 Conference, Orlando, Florida, 2000. 3.
[48] Paul C. Jorgensen. Software Testing: A Craftma'5 APProach, ZE. CRCPress, 1995. 5.
[49] CemKaner, JamesBaeh, BretPettiehord. Lessons Leamedin Software Testing. Wiley, 2001. 12.
[50] Zambelich K. TOtallyData—Driven Automated Testing. 1998, http: //www. sqa-test. com/waPerl. html.
[51] Kai-yUanCaia, Yong-ehaoLia, KeLiub. Optimal and ada Ptivetesting for software relj ability assessment Information and Software Teehnolog2004(46): 989-1000.
[52] TiloLinz, MatthiasDaigl. How to Automate Testing of GraPhieal User Interfaces. http: //www. imbus. de.
[53] Li Feng, Sheng ZhL, 19. Actioll-driven automation test framework for Graphieal. UserInterface (GUI) software testing. Autotestcon, 2007IEEE17-202007. 9: 22.
[54] Wtlst, GAModel for Sueeessful Software Testing Automation. STAR'99 East Conference, Orlando, Florida, 1999. 5.
[55] Pollner. An Advaneed Teehniquein Test Automation. STAREAST 2000 Conference, Orlando, Florida, 2000. 5.
[56] MichaelKelly. Choosing a test automation framework . http: //www. ibm. eom/develoPerworks/rationaFlibrary/591. html.
[57] Kanglin Li, Mengqi Wu. Effeetive Software Test Automation: Develo Pingan Automated Software Testing Tool, Sex, 2004.
[58] Ratlonal Funetional Tester User Guide. http: //www-306. ibm. eoln/software/rational/.
[59] Nagle C. Data Driven Test Antomation: For Rational Robot. V20001999-2000, DDEDoc Index.

高等学校计算机基础教育教材精选

书　名	书　号
Access 数据库基础教程　赵乃真	ISBN 978-7-302-12950-9
AutoCAD 2002 实用教程　唐嘉平	ISBN 978-7-302-05562-4
AutoCAD 2006 实用教程(第 2 版)　唐嘉平	ISBN 978-7-302-13603-3
AutoCAD 2007 中文版机械制图实例教程　蒋晓	ISBN 978-7-302-14965-1
AutoCAD 计算机绘图教程　李苏红	ISBN 978-7-302-10247-2
C++ 及 Windows 可视化程序设计　刘振安	ISBN 978-7-302-06786-3
C++ 及 Windows 可视化程序设计题解与实验指导　刘振安	ISBN 978-7-302-09409-8
C++ 语言基础教程(第 2 版)　吕凤翥	ISBN 978-7-302-13015-4
C++ 语言基础教程题解与上机指导(第 2 版)　吕凤翥	ISBN 978-7-302-15200-2
C++ 语言简明教程　吕凤翥	ISBN 978-7-302-15553-9
CATIA 实用教程　李学志	ISBN 978-7-302-07891-3
C 程序设计教程(第 2 版)　崔武子	ISBN 978-7-302-14955-2
C 程序设计辅导与实训　崔武子	ISBN 978-7-302-07674-2
C 程序设计试题精选　崔武子	ISBN 978-7-302-10760-6
C 语言程序设计　牛志成	ISBN 978-7-302-16562-0
PowerBuilder 数据库应用系统开发教程　崔巍	ISBN 978-7-302-10501-5
Pro/ENGINEER 基础建模与运动仿真教程　孙进平	ISBN 978-7-302-16145-5
SAS 编程技术教程　朱世武	ISBN 978-7-302-15949-0
SQL Server 2000 实用教程　范立南	ISBN 978-7-302-07937-8
Visual Basic 6.0 程序设计实用教程(第 2 版)　罗朝盛	ISBN 978-7-302-16153-0
Visual Basic 程序设计实验指导与习题　罗朝盛	ISBN 978-7-302-07796-1
Visual Basic 程序设计教程　刘天惠	ISBN 978-7-302-12435-1
Visual Basic 程序设计应用教程　王瑾德	ISBN 978-7-302-15602-4
Visual Basic 试题解析与实验指导　王瑾德	ISBN 978-7-302-15520-1
Visual Basic 数据库应用开发教程　徐安东	ISBN 978-7-302-13479-4
Visual C++ 6.0 实用教程(第 2 版)　杨永国	ISBN 978-7-302-15487-7
Visual FoxPro 程序设计　罗淑英	ISBN 978-7-302-13548-7
Visual FoxPro 数据库及面向对象程序设计基础　宋长龙	ISBN 978-7-302-15763-2
Visual LISP 程序设计(AutoCAD 2006)　李学志	ISBN 978-7-302-11924-1
Web 数据库技术　铁军	ISBN 978-7-302-08260-6
程序设计教程(Delphi)　姚普选	ISBN 978-7-302-08028-2
程序设计教程(Visual C++)　姚普选	ISBN 978-7-302-11134-4
大学计算机(应用基础·Windows 2000 环境)卢湘鸿	ISBN 978-7-302-10187-1
大学计算机基础　高敬阳	ISBN 978-7-302-11566-3
大学计算机基础实验指导　高敬阳	ISBN 978-7-302-11545-8
大学计算机基础　秦光洁	ISBN 978-7-302-15730-4
大学计算机基础实验指导与习题集　秦光洁	ISBN 978-7-302-16072-4
大学计算机基础　牛志成	ISBN 978-7-302-15485-3
大学计算机基础　訾秀玲	ISBN 978-7-302-13134-2

大学计算机基础习题与实验指导　訾秀玲　ISBN 978-7-302-14957-6
大学计算机基础教程(第 2 版)　张莉　ISBN 978-7-302-15953-7
大学计算机基础实验教程(第 2 版)　张莉　ISBN 978-7-302-16133-2
大学计算机基础实践教程(第 2 版)　王行恒　ISBN 978-7-302-18320-4
大学计算机技术应用　陈志云　ISBN 978-7-302-15641-3
大学计算机软件应用　王行恒　ISBN 978-7-302-14802-9
大学计算机应用基础　高光来　ISBN 978-7-302-13774-0
大学计算机应用基础上机指导与习题集　郝莉　ISBN 978-7-302-15495-2
大学计算机应用基础　王志强　ISBN 978-7-302-11790-2
大学计算机应用基础题解与实验指导　王志强　ISBN 978-7-302-11833-6
大学计算机应用基础教程　詹国华　ISBN 978-7-302-11483-3
大学计算机应用基础实验教程(修订版)　詹国华　ISBN 978-7-302-16070-0
大学计算机应用教程　韩文峰　ISBN 978-7-302-11805-3
大学信息技术(Linux 操作系统及其应用)　衷克定　ISBN 978-7-302-10558-9
电子商务网站建设教程(第 2 版)　赵祖荫　ISBN 978-7-302-16370-1
电子商务网站建设实验指导(第 2 版)　赵祖荫　ISBN 978-7-302-16530-9
多媒体技术及应用　王志强　ISBN 978-7-302-08183-8
多媒体技术及应用　付先平　ISBN 978-7-302-14831-9
多媒体应用与开发基础　史济民　ISBN 978-7-302-07018-4
基于 Linux 环境的计算机基础教程　吴华洋　ISBN 978-7-302-13547-0
基于开放平台的网页设计与编程(第 2 版)　程向前　ISBN 978-7-302-18377-8
计算机辅助工程制图　孙力红　ISBN 978-7-302-11236-5
计算机辅助设计与绘图(AutoCAD 2007 中文版)(第 2 版)　李学志　ISBN 978-7-302-15951-3
计算机软件技术及应用基础　冯萍　ISBN 978-7-302-07905-7
计算机图形图像处理技术与应用　何薇　ISBN 978-7-302-15676-5
计算机网络公共基础　史济民　ISBN 978-7-302-05358-3
计算机网络基础(第 2 版)　杨云江　ISBN 978-7-302-16107-3
计算机网络技术与设备　满文庆　ISBN 978-7-302-08351-1
计算机文化基础教程(第 3 版)　冯博琴　ISBN 978-7-302-19534-4
计算机文化基础教程实验指导与习题解答　冯博琴　ISBN 978-7-302-09637-5
计算机信息技术基础教程　杨平　ISBN 978-7-302-07108-2
计算机应用基础　林冬梅　ISBN 978-7-302-12282-1
计算机应用基础实验指导与题集　冉清　ISBN 978-7-302-12930-1
计算机应用基础题解与模拟试卷　徐士良　ISBN 978-7-302-14191-4
计算机应用基础教程　姜继忱　ISBN 978-7-302-18421-8
计算机硬件技术基础　李继灿　ISBN 978-7-302-14491-5
软件技术与程序设计(Visual FoxPro 版)　刘玉萍　ISBN 978-7-302-13317-9
数据库应用程序设计基础教程(Visual FoxPro)　周山芙　ISBN 978-7-302-09052-6
数据库应用程序设计基础教程(Visual FoxPro)题解与实验指导　黄京莲　ISBN 978-7-302-11710-0
数据库原理及应用(Access)(第 2 版)　姚普选　ISBN 978-7-302-13131-1
数据库原理及应用(Access)题解与实验指导(第 2 版)　姚普选　ISBN 978-7-302-18987-9
数值方法与计算机实现　徐士良　ISBN 978-7-302-11604-2
网络基础及 Internet 实用技术　姚永翘　ISBN 978-7-302-06488-6

网络基础与 Internet 应用　姚永翘　ISBN 978-7-302-13601-9
网络数据库技术与应用　何薇　ISBN 978-7-302-11759-9
网页设计创意与编程　魏善沛　ISBN 978-7-302-12415-3
网页设计创意与编程实验指导　魏善沛　ISBN 978-7-302-14711-4
网页设计与制作技术教程(第 2 版)　王传华　ISBN 978-7-302-15254-8
网页设计与制作教程(第 2 版)　杨选辉　ISBN 978-7-302-17820-0
网页设计与制作实验指导(第 2 版)　杨选辉　ISBN 978-7-302-17729-6
微型计算机原理与接口技术(第 2 版)　冯博琴　ISBN 978-7-302-15213-2
微型计算机原理与接口技术题解及实验指导(第 2 版)　吴宁　ISBN 978-7-302-16016-8
现代微型计算机原理与接口技术教程　杨文显　ISBN 978-7-302-12761-1
新编 16/32 位微型计算机原理及应用教学指导与习题详解　李继灿　ISBN 978-7-302-13396-4

读者意见反馈

亲爱的读者：

感谢您一直以来对清华版计算机教材的支持和爱护。为了今后为您提供更优秀的教材，请您抽出宝贵的时间来填写下面的意见反馈表，以便我们更好地对本教材做进一步改进。同时如果您在使用本教材的过程中遇到了什么问题，或者有什么好的建议，也请您来信告诉我们。

地址：北京市海淀区双清路学研大厦 A 座 602　　计算机与信息分社营销室 收
邮编：100084　　电子邮件：jsjjc@tup.tsinghua.edu.cn
电话：010-62770175-4608/4409　　邮购电话：010-62786544

教材名称：基于 RUP 的软件测试实践

ISBN：978-7-302-20247-9

个人资料

姓名：________ 年龄：________ 所在院校/专业：________

文化程度：________ 通信地址：________

联系电话：________ 电子信箱：________

您使用本书是作为： □指定教材 □选用教材 □辅导教材 □自学教材

您对本书封面设计的满意度：

□很满意 □满意 □一般 □不满意 改进建议________

您对本书印刷质量的满意度：

□很满意 □满意 □一般 □不满意 改进建议________

您对本书的总体满意度：

从语言质量角度看 □很满意 □满意 □一般 □不满意

从科技含量角度看 □很满意 □满意 □一般 □不满意

本书最令您满意的是：

□指导明确 □内容充实 □讲解详尽 □实例丰富

您认为本书在哪些地方应进行修改？（可附页）

____

____

您希望本书在哪些方面进行改进？（可附页）

____

____

电子教案支持

敬爱的教师：

为了配合本课程的教学需要，本教材配有配套的电子教案（素材），有需求的教师可以与我们联系，我们将向使用本教材进行教学的教师免费赠送电子教案（素材），希望有助于教学活动的开展。相关信息请拨打电话 010-62776969 或发送电子邮件至 jsjjc@tup.tsinghua.edu.cn 咨询，也可以到清华大学出版社主页（http://www.tup.com.cn 或 http://www.tup.tsinghua.edu.cn）上查询。